

ANOMALY-BASED NETWORK INTRUSION DETECTION
ENHANCEMENT BY PREDICTION THRESHOLD
ADAPTATION OF BINARY CLASSIFICATION MODELS

Amjad Mohamed Al Tobi

A Thesis Submitted for the Degree of PhD
at the
University of St Andrews



2018

Full metadata for this thesis is available in
St Andrews Research Repository
at:

<http://research-repository.st-andrews.ac.uk/>

Please use this identifier to cite or link to this thesis:

<http://hdl.handle.net/10023/17050>

This item is protected by original copyright

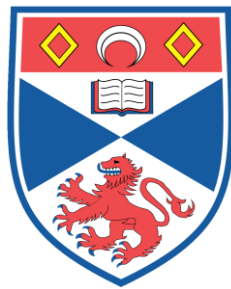
This item is licensed under a
Creative Commons Licence

<https://creativecommons.org/licenses/by/4.0/>

Anomaly-Based Network Intrusion Detection Enhancement by Prediction Threshold Adaptation of Binary Classification Models

Thesis by

Amjad Mohamed Al Tobi



University of
St Andrews

This thesis is submitted in partial fulfilment for the degree of
DOCTOR OF PHILOSOPHY (PhD)
at the UNIVERSITY OF ST ANDREWS

November 2018

Copyright © 2018 by Amjad M. Al Tobi

ABSTRACT

Network traffic exhibits a high level of variability over short periods of time. This variability impacts negatively on the performance (accuracy) of anomaly-based network Intrusion Detection Systems (IDS) that are built using predictive models in a batch-learning setup. This thesis investigates how adapting the discriminating threshold of model predictions, specifically to the evaluated traffic, improves the detection rates of these Intrusion Detection models. Specifically, this thesis studied the adaptability features of three well known **Machine Learning** algorithms: *C5.0*, *Random Forest*, and *Support Vector Machine*. The ability of these algorithms to adapt their prediction thresholds was assessed and analysed under different scenarios that simulated real world settings using the **prospective sampling** approach. A new dataset (STA2018) was generated for this thesis and used for the analysis.

This thesis has demonstrated empirically the importance of threshold adaptation in improving the accuracy of detection models when training and evaluation (test) traffic have different statistical properties. Further investigation was undertaken to analyse the effects of feature selection and data balancing processes on a model's accuracy when evaluation traffic with different significant features were used. The effects of threshold adaptation on reducing the accuracy degradation of these models was statistically analysed. The results showed that, of the three compared algorithms, Random Forest was the most adaptable and had the highest detection rates.

This thesis then extended the analysis to apply threshold adaptation on sampled traffic subsets, by using different *sample sizes*, *sampling strategies* and *label error rates*. This investigation showed the robustness of the Random Forest algorithm in identifying the best threshold. The Random Forest algorithm only needed a sample that was 0.05% of the original evaluation traffic to identify a discriminating threshold with an overall accuracy rate of nearly 90% of the optimal threshold.

DECLARATION

Candidate's Declarations

I, AMJAD MOHAMED HAMDAN AL TOBI, do hereby certify that this thesis, submitted for the degree of PhD, which is approximately 59,400 words in length, has been written by me, and that it is the record of work carried out by me, or principally by myself in collaboration with others as acknowledged, and that it has not been submitted in any previous application for any degree.

I was admitted as a research student at the UNIVERSITY OF ST ANDREWS in September 2014.

I, AMJAD MOHAMED HAMDAN AL TOBI, received assistance in the writing of this thesis in respect of grammar and spelling, which was provided by UNA BARTLEY (freelance proofreader).

I received funding from an organisation or institution and have acknowledged the funder(s) in the full text of my thesis.

Signature of Candidate:

Date: 20 November 2018

Supervisor's Declaration

I, DR. ISHBEL M. DUNCAN, hereby certify that the candidate has fulfilled the conditions of the Resolution and Regulations appropriate for the degree of PhD in the University of St Andrews and that the candidate is qualified to submit this thesis in application for that degree.

Signature of Supervisor:

Date: 20 November 2018

PERMISSION FOR PUBLICATION

In submitting this thesis to the UNIVERSITY OF ST ANDREWS we understand that we are giving permission for it to be made available for use in accordance with the regulations of the UNIVERSITY LIBRARY for the time being in force, subject to any copyright vested in the work not being affected thereby. We also understand, unless exempt by an award of an embargo as requested below, that the title and the abstract will be published, and that a copy of the work may be made and supplied to any bona fide library or research worker, that this thesis will be electronically accessible for personal or research use and that the library has the right to migrate this thesis into new electronic forms as required to ensure continued access to the thesis.

I, AMJAD MOHAMED HAMDAN AL TOBI, confirm that my thesis does not contain any third-party material that requires copyright clearance.

The following is an agreed request by candidate and supervisor regarding the publication of this thesis:

Printed copy

No embargo on print copy.

Electronic copy

No embargo on electronic copy.

Signature of Candidate:

Date: 20 November 2018

Signature of Supervisor:

Date: 20 November 2018

UNDERPINNING RESEARCH DATA OR DIGITAL OUTPUTS

Candidate's Declarations

I, AMJAD MOHAMED HAMDAN AL TOBI, understand that by declaring that I have original research data or digital outputs, I should make every effort in meeting the University's and research funders' requirements on the deposit and sharing of research data or research digital outputs.

Signature of Candidate:

Date: 20 November 2018

Research data underpinning this thesis are available at

Full STA2018 dataset:

[<http://dx.doi.org/10.17630/c5f31888-9db5-4ac0-a990-3fd17dcfe865>].

Digital outputs underpinning this thesis are available at

Experiment codes: <https://github.com/elud074/phdExperiments>

STA2018 generation codes: <https://github.com/elud074/STA2018>

Permission for publication of underpinning research data or digital outputs

We understand that for any original research data or digital outputs which are deposited, we are giving permission for them to be made available for use in accordance with the requirements of the University and research funders, for the time being in force.

We also understand that the title and the description will be published, and that the underpinning research data or digital outputs will be electronically accessible for use in accordance with the license specified at the point of deposit, unless exempt by award of an embargo as requested below.

The following is an agreed request by candidate and supervisor regarding the publication of underpinning research data or digital outputs:

Embargo on all of electronic files for a period of 2 years on the following ground(s):

- Publication would preclude future publication

Supporting statement for embargo request

I and my supervisor planning to publish the dataset in a journal.

Signature of Candidate:

Date: 20 November 2018

Signature of Supervisor:

Date: 20 November 2018

FUNDING

This research was supported and funded by the GOVERNMENT OF THE SULTANATE OF OMAN represented by the MINISTRY OF HIGHER EDUCATION and the SULTAN QABOOS UNIVERSITY.

NOTE ON WEB RESOURCES

Where Web based resources are cited via footnotes or references, these resources were checked and verified and were accessible upon the date of the thesis' submission. However, due to the nature of such resources their longevity cannot be guaranteed. Such references have only been used where citations to more traditional Peer-Reviewed published material were not possible.

ACKNOWLEDGEMENTS

All praise be to ALMIGHTY ALLAH, the GRACIOUS, the MERCIFUL and the Lord of all worlds, who granted me the will to pursue my studies and blessed me to accomplish this dream. All deepest gratitude to his slave and messenger MOHAMMED (*Peace Be Upon Him*).

I would like, first and foremost, to convey my gratefulness to my supervisor, DR. ISHBEL M. DUNCAN for accepting to supervise my PhD research, as well as for her continuous guidance, mentoring and support. To all those weekly meetings, encouragement to question science and challenge myself even more, I say from the bottom of my heart: *Thank you*.

My sincere gratitude is extended to my second supervisor, PROF. SALEEM BATTI for all the wisdom he provided and the valuable remarks that contributed in refining this thesis.

I would like to express my appreciation to my friends who I have been privileged to be surrounded by. Those who always supported me and provided me with required encouragement to keep pushing myself beyond the limits. My special thanks to my esteemed fellow PhD students; YASIR ALGUWAIFLI, KHAWAR SHEHZAD, HUSSEIN BAKRI, DR. PERCY PEREZ ARUNI, and HAIFA AL NASSERI.

I also would like to thank all of the members in the Machine Learning Reading (MLR) group, whom I have always gained from their deep and enlightening discussions. My special thanks to DR. LEI FANG, who has formed the group and to all his help in understanding the complex math behind this domain. Thanks to DR. MARK-JAN NEDERHOF for his deep insights and efforts in sharing his understandings with the rest of the group and for the valuable time he devoted in providing me with important feedbacks. My sincere thanks are extend as well to DR. JUAN YE for her valuable comments that helped improve this thesis.

Many thanks are given to everyone in the following units in the UNIVERSITY OF ST ANDREWS whom I might not know them in person, but their commitment and efforts have made my experience throughout my PhD the most pleasant and fruitful journey. Thanks to the administration of the Computer Science School; the IT support team (FixIT); the Centre for

Academic, Professional and Organisational Development (CAPOD); iELS team; Systems Research Group (SRG).

To my family, all my gratefulness and gratitude are extended. All my warm thanks to my beloved mother who always remembered me in her prayers. To my father, thank you for your endless encouragement that ignited the passion in my heart to seek knowledge and wisdom. To both of you I dedicate this humble work.

To the mate of my soul, my treasured and beloved wife, JALILA K. AL TOBI, thank you for lighting the candle in dark moments to see the path and being on my side all the way to the end of the tunnel.

Last but not the least, my special thanks and gratefulness are extended to my country who gave me the opportunity to continue my further studies and pursuit my dream to do my PhD. This research was supported and funded by the GOVERNMENT OF THE SULTANATE OF OMAN represented by the MINISTRY OF HIGHER EDUCATION, whom without their scholarship this thesis would not have existed.

AMJAD M. AL TOBI
The University of St Andrews
November 2018

CONTENTS

Abstract.....	i
Declaration.....	iii
Permission for Publication	v
Underpinning Research Data or Digital Outputs.....	vii
Acknowledgements	xi
Contents	xiii
List of Figures.....	xvii
List of Tables	xxiii
Listings	xxvii
Chapter 1: Introduction	1
1.1 Problem Statement.....	2
1.2 Motivation.....	4
1.3 Scope of the Research.....	5
1.4 Research Hypothesis and Questions	7
1.5 Research Approach	9
1.6 Contributions	9
1.7 Research Output.....	12
1.8 Thesis Structure	12
Chapter 2: Literature Review and State of the Art.....	15
2.1 Intrusion Detection (ID).....	15
2.2 Datasets	25
2.3 Evaluation of Intrusion Detection Systems	36
2.4 Related Work and Research Gaps.....	46
2.5 Summary	58
Chapter 3: Experimental Overview	59
3.1 Overview of Experiments	60
3.2 Overview of Classification/Machine Learning algorithms.....	62
3.3 Methods Used for Analysis.....	73
3.4 Limitations	82
3.5 Summary.....	83
Chapter 4: Adaptive Cutoff (Threshold) for Prediction Models	85

4.1	Problem Statement	85
4.2	Proposed Solution.....	86
4.3	Datasets	88
4.4	Experimental Setting	92
4.5	Results and Discussion.....	94
4.6	Limitations.....	107
4.7	Summary	108
Chapter 5: UNB ISCX 2012 Dataset Transformation		109
5.1	ISCX2012 Dataset Description	110
5.2	Transformation Process.....	112
5.3	Details of Validation and Labelling Phase	122
5.4	Server Specifications.....	137
5.5	Limitations.....	138
5.6	Summary	139
Chapter 6: Effect of Feature Selection and Data Balance on Adaptive Cutoff for Network Intrusion Detection.....		143
6.1	Introduction	143
6.2	Proposed Solution.....	149
6.3	Datasets	152
6.4	Experimental Setting	159
6.5	Results and Discussion.....	162
6.6	Limitations.....	175
6.7	Summary	177
Chapter 7: Cutoff Selection Based on Evaluating a Subset of the Test Data		179
7.1	Introduction	179
7.2	Proposed Solution.....	182
7.3	Experimental Setting	184
7.4	Results and Discussion.....	187
7.5	Limitations.....	207
7.6	Summary	209
Chapter 8: Conclusion		211
8.1	Main Findings.....	212
8.2	Future Work	216
8.3	Reflective/Closing Remarks.....	220
Appendix (A) Results of Chapter 4 (First) Experiment.....		221
A.1.	C5.0	222
A.2.	Random Forest	223

A.3. Support Vector Machine (SVM)	225
Appendix (B) Results of Chapter 6 (Second) Experiment.....	227
B.1. Selected Features	227
B.2. Models Results.....	233
Appendix (C) Results of Chapter 7 (Third) Experiment.....	243
C.1. Results of Every Day	243
C.2. Models GAR Plots	262
Appendix (D) Feature Descriptions of STA2018 Dataset.....	267
D.1. Basic Features	268
D.2. Connection-Based Features	275
D.3. Time-Based Features	286
D.4. Class Feature.....	295
References	297

LIST OF FIGURES

Figure 2.1: Plot of the confusion matrices for dummy data with 100 instances (negative (class 0) = 10 instances and positive (class 1) = 90 instances) at different cutoff (threshold) values. (a) Cutoff (threshold) is 0. (b) Cutoff (threshold) is 0.5. (c) Cutoff (threshold) is 1.	44
Figure 2.2: ROC curves for two dummy datasets with a different number of instances; D_{100} has 100 instances while D_{1000} has 1000 instances. For each dataset 10% of its instances are negative (class 0) while the remaining 90% are positive (class 1).	45
Figure 3.1: Schematic overview of the experiments undertaken.....	61
Figure 3.2: Example of a decision tree of dummy network traffic data with two classes {attack and normal}. (a) Returns the class label. (b) Returns the probability of classes.	63
Figure 3.3: Main phases of Random Forest algorithm	68
Figure 3.4: Example of SVM on two dimensional dummy data, where a_1, a_2, a_3 : input data points (vectors), w : normal vector to the hyperplane (weight vector) and b : bias. (a) Perfectly separable dataset. (b) Dataset separation with soft margin.	69
Figure 3.5: Data transformation from 2D in input space to 3D in the feature space using a kernel function. The figure was reproduced with some modification from Statnikov et. al [305].....	72
Figure 3.6: K-folds Cross-Validation process.	80
Figure 3.7: Friedman’s Test computation and interpretation	81
Figure 4.1: Command used to generate File 1 of SEA dataset.....	91
Figure 4.2: Command used to generate File 1 of AGR dataset.....	92
Figure 4.3: The experiments’ phases diagram.....	93
Figure 4.4: G-Mean accuracy curves of the 10 runs of the 10-folds Cross-Validation experiments for the three datasets (gureKDD, SEA and AGR) using three classification algorithms. (a) C5.0. (b) Random Forest. (c) SVM.....	96
Figure 4.5: Critical differences plot of the pairwise Nemenyi comparison test for the full datasets 10-folds Cross-Validation experiment.	98
Figure 4.6: G-Mean Accuracy Curves for the C5.0 Algorithm (for gureKDD data see Table A.1, SEA data see Table A.2 and AGR data see Table A.3).....	100
Figure 4.7: G-Mean Accuracy Curves for Random Forest Algorithm (for gureKDD data see Table A.4, SEA data see Table A.5 and AGR data see Table A.6).....	102
Figure 4.8: G-Mean Accuracy Curves for the SVM Algorithm (for gureKDD data see Table A.7, SEA data see Table A.8 and AGR data see Table A.9).....	104

Figure 4.9: Critical differences plot of the pairwise Nemenyi comparison test for the cutoff (threshold) adaptation experiment.	106
Figure 5.1: Dataset preparation phases	113
Figure 5.2: Onut’s Feature Classification Schema [15].....	117
Figure 5.3: ISCX2012 number of class connections for each simulation day.....	119
Figure 5.4: Generation of synthetic instances using the SMOTE algorithm	120
Figure 5.5: Features in the flow files (XML) in all analyses.	124
Figure 5.6: Connection matching by mapping keys	127
Figure 6.1: Illustration of the information system extension with fake features/variable.	147
Figure 6.2: Figures adapted from Lex et al. [372]. (a) Examples of slicing and aggregation, including aggregation ‘by degree’ which is used in this chapter. (b) Example of a set relationship encoded by columns from the matrix, where the sets that contribute to every exclusive intersection are represented by filled dark circles connected by a line.....	156
Figure 6.3: Plots of feature sets’ intersections using the MDA measure (a) Day Features’ intersections using the MDA measure on the original (imbalanced) data. (b) Day Features’ intersections using the $MDA_{Balance}$ measure on the balanced data.	157
Figure 6.4: Plots of feature sets’ intersections using the MDG measure (a) Day Features’ intersections using the MDG measure on the original (imbalanced) data. (b) Day Features’ intersections using the $MDG_{Balance}$ measure on the balanced data.	158
Figure 6.5: Experimental phases diagram.....	160
Figure 6.6: Graphical illustration of pairwise comparisons from the Friedman Test results for different threshold effects (optimal or adaptive cutoff) after applying the Nemenyi test (95% confidence level) (a) Nemenyi test for different thresholds. (b) Nemenyi test for different algorithms. (c) Nemenyi test for different feature sets. (d) Nemenyi test for different training data balances.....	165
Figure 6.7: Nemenyi test (95% confidence level) on the C5.0 algorithm models using different feature sets and different data balances after applying the adaptive cutoff approach.....	168
Figure 6.8: Nemenyi test (95% confidence level) on the RF algorithm models using different feature sets and different data balances after applying the adaptive cutoff approach.....	170
Figure 6.9: Nemenyi test (95% confidence level) on SVM algorithm models using different feature sets and different data balances after applying the adaptive cutoff approach.....	172
Figure 6.10: Comparison plot of the average performance of every C5.0, RF and SVM model for every feature set and data balance combination.	174
Figure 7.1: Example comparing fixed threshold (solid black vertical line - Thr0.5) with adapted prediction thresholds (maroon, blue and orange vertical lines).....	180

Figure 7.2: Example comparing two predictions (maroon and blue). Vertical solid lines represent the optimal threshold of these predictions and the vertical dotted lines represent the sample cutoffs.....	185
Figure 7.3: Median of G-Mean Accuracy Ratios (GAR) of the C5.0 models predictions under different sampling strategies (number of bins), sample sizes and error rates.....	189
Figure 7.4: Average number of unique thresholds for the predictions of C5.0 models, and their ranges.....	190
Figure 7.5: Results of multiple Nemenyi tests (95% confidence level) on different sampling strategies (B_1 , B_{10} , B_{20} , B_{50} and B_{100}) using the C5.0 predictions under different error rates (0%, 1%, 5% and 10%).	192
Figure 7.6: Results of multiple Nemenyi tests (95% confidence level) of different sampling size (10% to 0.0001%) using the C5.0 predictions under different error rates (0%, 1%, 5% and 10%).	193
Figure 7.7: Median of G-Mean Accuracy Ratios (GAR) of the RF models predictions under different sampling strategies (number of bins), sample sizes and error rates.....	194
Figure 7.8: Average number of unique thresholds for the predictions of the RF models, and their ranges.....	195
Figure 7.9: Results of multiple Nemenyi tests (95% confidence level) on different sampling strategies (B_1 , B_{10} , B_{20} , B_{50} and B_{100}) using the RF predictions under different error rates (0%, 1%, 5% and 10%).	196
Figure 7.10: Results of multiple Nemenyi tests (95% confidence level) on different sampling sizes (10% to 0.0001%) using RF predictions under different error rates (0%, 1%, 5% and 10%).	198
Figure 7.11: Median of G-Mean Accuracy Ratios (GAR) of the predictions of the SVM models under different sampling strategies (number of bins), sample sizes and error rates.	199
Figure 7.12: Average number of unique thresholds for the predictions of the SVM models and their ranges.....	200
Figure 7.13: Results of multiple Nemenyi tests (95% confidence level) on different sampling strategies (B_1 , B_{10} , B_{20} , B_{50} and B_{100}) using the SVM predictions under different error rates (0%, 1%, 5% and 10%).	202
Figure 7.14: Results of multiple Nemenyi tests (95% confidence level) on different sampling sizes (10% to 0.0001%) using the SVM predictions under different error rates (0%, 1%, 5% and 10%).	204
Figure 7.15: Illustrative plots of the effect of erroneous sample labels on threshold shift (a) C5.0 predictions (b) RF and SVM predictions.....	205
Figure 7.16: Critical Difference plots for the different ML algorithm samples under different error rates.	207
Figure B.1: Comparison plot of the performance of C5.0 models (G-Mean Accuracy) for every training day in the STA2018 dataset between the optimal (CV) and adaptive cutoffs.....	235

Figure B.2: Comparison plot of the performance of RF models (G-Mean Accuracy) for every training day in the STA2018 dataset between the optimal (CV) and adaptive cutoffs.	238
Figure B.3: Comparison plot of the performance of SVM models (G-Mean Accuracy) for every training day in the STA2018 dataset between the optimal (CV) and adaptive cutoffs.	241
Figure C.1: Day 2 (12/Jun) results for the C5.0 models. (a) Plots of G-Mean Accuracy Ratio (GAR) of Day 6 models at an error rate of 0%. (b) GAR plots at an error rate of 1%. (c) GAR plots at an error rate of 5%. (d) GAR plots at an error rate of 10%.	244
Figure C.2: Day 3 (13/Jun) results for the C5.0 models. (a) Plots of G-Mean Accuracy Ratio (GAR) of Day 6 models at an error rate of 0%. (b) GAR plots at an error rate of 1%. (c) GAR plots at an error rate of 5%. (d) GAR plots at an error rate of 10%.	245
Figure C.3: Day 4 (14/Jun) results for the C5.0 models. (a) Plots of G-Mean Accuracy Ratio (GAR) of Day 6 models at an error rate of 0%. (b) GAR plots at an error rate of 1%. (c) GAR plots at an error rate of 5%. (d) GAR plots at an error rate of 10%.	246
Figure C.4: Day 5 (15/Jun) results for the C5.0 models. (a) Plots of G-Mean Accuracy Ratio (GAR) of Day 6 models at an error rate of 0%. (b) GAR plots at an error rate of 1%. (c) GAR plots at an error rate of 5%. (d) GAR plots at an error rate of 10%.	247
Figure C.5: Day 6 (16/Jun) results for the C5.0 models. (a) Plots of G-Mean Accuracy Ratio (GAR) of Day 6 models at an error rate of 0%. (b) GAR plots at an error rate of 1%. (c) GAR plots at an error rate of 5%. (d) GAR plots at an error rate of 10%.	248
Figure C.6: Day 7 (17/Jun) results for the C5.0 models. (a) Plots of G-Mean Accuracy Ratio (GAR) of Day 6 models at an error rate of 0%. (b) GAR plots at an error rate of 1%. (c) GAR plots at an error rate of 5%. (d) GAR plots at an error rate of 10%.	249
Figure C.7: Day 2 (12/Jun) results for the RF models. (a) Plots of G-Mean Accuracy Ratio (GAR) of Day 6 models at an error rate of 0%. (b) GAR plots at an error rate of 1%. (c) GAR plots at an error rate of 5%. (d) GAR plots at an error rate of 10%.	250
Figure C.8: Day 3 (13/Jun) results for the RF models. (a) Plots of G-Mean Accuracy Ratio (GAR) of Day 6 models at an error rate of 0%. (b) GAR plots at an error rate of 1%. (c) GAR plots at an error rate of 5%. (d) GAR plots at an error rate of 10%.	251
Figure C.9: Day 4 (14/Jun) results for the RF models. (a) Plots of G-Mean Accuracy Ratio (GAR) of Day 6 models at an error rate of 0%. (b) GAR plots at an error rate of 1%. (c) GAR plots at an error rate of 5%. (d) GAR plots at an error rate of 10%.	252
Figure C.10: Day 5 (15/Jun) results for the RF models. (a) Plots of G-Mean Accuracy Ratio (GAR) of Day 6 models at an error rate of 0%. (b) GAR plots at an error	

rate of 1%. (c) GAR plots at an error rate of 5%. (d) GAR plots at an error rate of 10%.....	253
Figure C.11: Day 6 (16/Jun) results for the RF models. (a) Plots of G-Mean Accuracy Ratio (GAR) of Day 6 models at an error rate of 0%. (b) GAR plots at an error rate of 1%. (c) GAR plots at an error rate of 5%. (d) GAR plots at an error rate of 10%.....	254
Figure C.12: Day 7 (17/Jun) results for the RF models. (a) Plots of G-Mean Accuracy Ratio (GAR) of Day 6 models at an error rate of 0%. (b) GAR plots at an error rate of 1%. (c) GAR plots at an error rate of 5%. (d) GAR plots at an error rate of 10%.....	255
Figure C.13: Day 2 (12/Jun) results for the SVM models. (a) Plots of G-Mean Accuracy Ratio (GAR) of Day 6 models at an error rate of 0%. (b) GAR plots at an error rate of 1%. (c) GAR plots at an error rate of 5%. (d) GAR plots at an error rate of 10%.....	256
Figure C.14: Day 3 (13/Jun) results for the SVM models. (a) Plots of G-Mean Accuracy Ratio (GAR) of Day 6 models at an error rate of 0%. (b) GAR plots at an error rate of 1%. (c) GAR plots at an error rate of 5%. (d) GAR plots at an error rate of 10%.....	257
Figure C.15: Day 4 (14/Jun) results for the SVM models. (a) Plots of G-Mean Accuracy Ratio (GAR) of Day 6 models at an error rate of 0%. (b) GAR plots at an error rate of 1%. (c) GAR plots at an error rate of 5%. (d) GAR plots at an error rate of 10%.....	258
Figure C.16: Day 5 (15/Jun) results for the SVM models. (a) Plots of G-Mean Accuracy Ratio (GAR) of Day 6 models at an error rate of 0%. (b) GAR plots at an error rate of 1%. (c) GAR plots at an error rate of 5%. (d) GAR plots at an error rate of 10%.....	259
Figure C.17: Day 6 (16/Jun) results for the SVM models. (a) Plots of G-Mean Accuracy Ratio (GAR) of Day 6 models at an error rate of 0%. (b) GAR plots at an error rate of 1%. (c) GAR plots at an error rate of 5%. (d) GAR plots at an error rate of 10%.....	260
Figure C.18: Day 7 (17/Jun) results for the SVM models. (a) Plots of G-Mean Accuracy Ratio (GAR) of Day 6 models at an error rate of 0%. (b) GAR plots at an error rate of 1%. (c) GAR plots at an error rate of 5%. (d) GAR plots at an error rate of 10%.....	261
Figure C.19: Medians of G-Mean Accuracy Ratios (GAR) of C5.0 models for every feature set and data balance type combination. (a) Medians at an error rate of 0%. (b) Medians at error rate of 1%. (c) Medians at an error rate of 5%. (d) Medians at an error rate of 10%.	263
Figure C.20: Medians of G-Mean Accuracy Ratios (GAR) of RF models for every feature set and data balance type combination. (a) Medians at an error rate of 0%. (b) Medians at error rate of 1%. (c) Medians at an error rate of 5%. (d) Medians at an error rate of 10%.	264
Figure C.21: Medians of G-Mean Accuracy Ratios (GAR) of SVM models for every features set and data balance type combination. (a) Medians at an error rate	

of 0%. (b) Medians at error rate of 1%. (c) Medians at an error rate of 5%.
(d) Medians at an error rate of 10%. 265

LIST OF TABLES

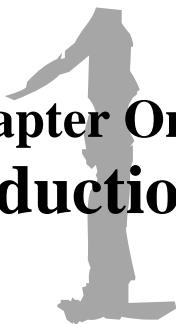
Table 2.1: Confusion Matrices. (a) Confusion matrix for binary classification which is a special case of multiple classes. (b) Confusion matrix for multiple classes.....	39
Table 2.2: Confusion matrices of dummy data with 100 instances (negative = 10 instances and positive = 90 instances) at different cutoff (threshold) values. (a) Cutoff (threshold) is 0. (b) Cutoff (threshold) is 0.5. (c) Cutoff (threshold) is 1.....	39
Table 3.1: Factors and levels of every experiment.....	75
Table 3.2: Runtime in seconds between kernel and linear SVM setups on a 10% subset of Day 2 (12/Jun) and Day 3 (13/Jun) of the STA2018 dataset.....	79
Table 3.3: Class labels mapping for SVM algorithm.....	79
Table 4.1: Number of connection classes in every file in the gureKDD dataset.....	90
Table 4.2: Number of instances' classes in every file in the SEA dataset	91
Table 4.3: Number of instances' classes in every file in the AGR dataset.....	92
Table 4.4: Average model performances (AUC and G-Mean Accuracy), the average Optimal Cutoff value (at which maximum G-Mean Accuracy was reached) and their standard deviation of the 10-folds Cross-Validation (10 repetitions)	96
Table 4.5: Model G-Mean accuracies arrangements of the 10-folds Cross-Validation on the full dataset for Friedman's test.	97
Table 4.6: Results of the pairwise Nemenyi comparison test for the full datasets 10-folds Cross-Validation experiment.	97
Table 4.7: Algorithms' medians and mean ranks for the full datasets 10-folds Cross-Validation experiment.	98
Table 4.8: Models G-Mean accuracies arrangements for Friedman's test for phase two of the experiments.....	105
Table 4.9: Results of the pairwise Nemenyi comparison test for the cutoff (threshold) adaptation experiment.....	105
Table 4.10: Algorithms' medians and mean ranks based on threshold adaptation effect.	106
Table 5.1: UNB ISCX 2012 dataset files.	111
Table 5.2: Packet counts comparison for all Bro, PCAP and XML files	116
Table 5.3: Basic features used in Onut's schema to extend the features set	118
Table 5.4: Indexes of eliminated features.....	122
Table 5.5: Example of total packets in every communication direction	123
Table 5.6: IPv6 address as in PCAP files	128

Table 5.7: Number of duplicate connections	130
Table 5.8: Comparison of number of labels between authored paper, XML files, and PCAP files.....	131
Table 5.9: Example of a split connection.....	132
Table 5.10: Example of XML connections	133
Table 5.11: Example of number of XML packet for each communication direction.....	133
Table 5.12: Number of non-matched flows in the XML Files.....	134
Table 5.13: Number of unique IP:PORTs in the PCAP and XML files	134
Table 5.14: Example of connections with the wrong direction	135
Table 5.15: Example of a UDP Connection’s wrong direction	135
Table 5.16: Number of connections with zero source packets	136
Table 5.17: Number of connections with the wrong number of packets	137
Table 6.1: Categorical (factor) features eliminated from the feature importance evaluation phase.....	148
Table 6.2: Number of classes of instances for each day’s file of the STA2018 dataset.	153
Table 6.3: Number of shared selected features for each two day pair after using the Mean Decrease of Accuracy (MDA) importance measure on the original and balanced versions of the data.	154
Table 6.4: Number of shared selected features for each two day pair using the Mean Decrease Gini (MDG) importance measure on the original and balanced versions of the data.....	154
Table 6.5: Number of features in all intersection areas (slices) of an aggregation degree for every feature set and their proportions in relation to the total number of unique features within each feature set.	159
Table 6.6: Model’s average performance for different ML algorithms, feature sets and data balances.	167
Table 7.1: Number of sampled instances for each sample size used in these experiments. ...	187
Table A.1: C5.0 model’s performance on gureKDD dataset with various effects (before and after threshold adaptation). MT (Model optimal Threshold); CA (Cross-validation G-Mean Accuracy); MA (Model threshold G-Mean Accuracy); FT (File optimal Threshold); FA (File threshold G-Mean Accuracy). Shaded cells are the 10-folds Cross-Validation results from the model generation phase.	222
Table A.2: C5.0 model’s performance on SEA dataset with various effects (before and after threshold adaptation). MT (Model optimal Threshold); CA (Cross-validation G-Mean Accuracy); MA (Model threshold G-Mean Accuracy); FT (File optimal Threshold); FA (File threshold G-Mean Accuracy). Shaded cells are the 10-folds Cross-Validation results from the model generation phase.	222
Table A.3: C5.0 model’s performance on AGR dataset with various effects (before and after threshold adaptation). MT (Model optimal Threshold); CA (Cross-validation G-Mean Accuracy); MA (Model threshold G-Mean Accuracy); FT	

(File optimal Threshold); FA (File threshold G-Mean Accuracy). Shaded cells are the 10-folds Cross-Validation results from the model generation phase.....	223
Table A.4: Random Forest (RF) model's performance on gureKDD dataset with various effects (before and after threshold adaptation). MT (Model optimal Threshold); CA (Cross-validation G-Mean Accuracy); MA (Model threshold G-Mean Accuracy); FT (File optimal Threshold); FA (File threshold G-Mean Accuracy). Shaded cells are the 10-folds Cross-Validation results from the model generation phase.	223
Table A.5: Random Forest (RF) model's performance on SEA dataset with various effects (before and after threshold adaptation). MT (Model optimal Threshold); CA (Cross-validation G-Mean Accuracy); MA (Model threshold G-Mean Accuracy); FT (File optimal Threshold); FA (File threshold G-Mean Accuracy). Shaded cells are the 10-folds Cross-Validation results from the model generation phase.	224
Table A.6: Random Forest (RF) model's performance on AGR dataset with various effects (before and after threshold adaptation). MT (Model optimal Threshold); CA (Cross-validation G-Mean Accuracy); MA (Model threshold G-Mean Accuracy); FT (File optimal Threshold); FA (File threshold G-Mean Accuracy). Shaded cells are the 10-folds Cross-Validation results from the model generation phase.	224
Table A.7: SVM model's performance on gureKDD dataset with various effects (before and after threshold adaptation). MT (Model optimal Threshold); CA (Cross-validation G-Mean Accuracy); MA (Model threshold G-Mean Accuracy); FT (File optimal Threshold); FA (File threshold G-Mean Accuracy). Shaded cells are the 10-folds Cross-Validation results from the model generation phase.....	225
Table A.8: SVM model's performance on SEA dataset with various effects (before and after threshold adaptation). MT (Model optimal Threshold); CA (Cross-validation G-Mean Accuracy); MA (Model threshold G-Mean Accuracy); FT (File optimal Threshold); FA (File threshold G-Mean Accuracy). Shaded cells are the 10-folds Cross-Validation results from the model generation phase.....	225
Table A.9: SVM model's performance on AGR dataset with various effects (before and after threshold adaptation). MT (Model optimal Threshold); CA (Cross-validation G-Mean Accuracy); MA (Model threshold G-Mean Accuracy); FT (File optimal Threshold); FA (File threshold G-Mean Accuracy). Shaded cells are the 10-folds Cross-Validation results from the model generation phase.....	226
Table B.1: The performance of models (G-Mean Accuracy) for the original and adapted cutoff (threshold) for the C5.0 algorithm.	234
Table B.2: Performance of the models (G-Mean Accuracy) for the original and adapted cutoffs (threshold) for the Random Forest (RF) algorithm.	237
Table B.3: The performance of models (G-Mean Accuracy) for the original and adapted cutoffs (thresholds) for SVM algorithm.	240
Table C.1: Figures map of the results of the third experiment.	243

LISTINGS

Listing 3.1: Pseudocode of main stages of Random Forest algorithm	66
Listing 4.1: Pseudocode of threshold adaptation process and the selections of the optimal threshold for the evaluated data.....	88
Listing 5.1: Pseudo code of connection labelling through mapping connections between PCAP and XML.....	127
Listing 5.2: Pseudo code of comparing number of packets between PCAP and XML connections	133
Listing 6.1: Pseudo code of the feature selection function.....	149
Listing 6.2: Pseudo code of the experimental phases.....	161
Listing 7.1: Pseudo code of the experiments run for the results of each ML algorithm.	186



Chapter One Introduction

Computer security, now called cyber security, has been a focus of research since the beginning of the digital computer age. Numerous research papers and applications have been written and developed proposing solutions to combat threats and to protect information systems. With the introduction of the Internet, more services have been provided remotely, which has led to increased dependence of users, business and governments on third party services. This dependence has promoted new and more sophisticated types of attacks, making the task of protecting and securing systems more challenging. As a consequence, the Internet has become a digital war zone for politicians and governments, as well as business competitors. Persistent digital warfare and/or information gathering imposes a greater risk than ever on systems and services, including individuals' private information.

Many authors including Cherdantseva and Hilton [2] have identified a set of key goals for cybersecurity that should be met by all systems: *confidentiality, integrity, availability, accountability, non-repudiation, auditability, authenticity & trustworthiness, privacy* and *correctness* [3-7]. However, every day these stated security goals are flagrantly violated by breaches and security incidents which raises questions about the capability of existing security systems.

Intrusion Detection Systems (IDS) are one of the many tools used in the cyber security field. Their main purpose, as an essential line of defence, is to detect security attacks targeting the critical networks, systems or data that they monitor. They aim to detect and report any violation by an external intruder or system insider, of the security goals highlighted above.

1.1 Problem Statement

Everyday advancements in technology, such as mobile devices, cloud computing and the Internet of Things (IoT), bring with them novel challenges and threats. As the majority of these technologies provide their services over communication networks, new challenges have emerged. For example, very large amounts of data are generated and exchanged across these networks that require faster processing. In addition, with various kinds of services sharing the same communication media, traffic diversity has become another challenge for detecting and profiling threats. Traffic encryption has added yet another layer of sophistication to any analysis task. Such challenges, and many more, have posed an immense burden on security analysts in analysing traffic and identifying threats in order to develop the right counter measures. As a result, researchers have aimed at exploring new tools, techniques and strategies to address such limitations.

Artificial Intelligence (AI), Machine Learning (ML) and Data Mining (DM) methods are some of the key research topics currently being explored to address some of the many cyber security requirements, particularly in the area of anomaly-based Intrusion Detection (ID). These methods became more pervasive than before in real world applications due to the advancement in technology, and are now being used in many different domains such as autonomous vehicles, directing advertising, healthcare, product recommendations, stock markets and speech or face recognition. The use of these methods aims to address the many limitations in human capabilities and conventional technologies in handling the massive amounts and existing diversity of digital data.

One of the most pressing challenges in this domain is the traffic evolution over time and the capability of these methods to adapt to such changes. This is because network traffic is not stable through time, due to changes in services, where new services are introduced and old ones are removed, as well as changes to users and their behaviours. Although many recent studies and solutions have been conducted and proposed, that have made remarkable progress in addressing these concerns, plenty of work and opportunities still remain.

Ever evolving traffic makes the process of building ID models a particularly challenging task as learning all possible variations of traffic patterns for all different kinds of traffic and users is an impossible quest. Therefore, there is a pressing need to make intelligent detection methods adaptable to traffic variability.

In a typical (batch-based) scenario, a network-based anomaly ID model would be built to protect specific environment from attackers. The model building phase would require some training data that were previously captured from old traffic to generate the ID model, which would be tuned and set to detect anomalous behaviours. However, as such a model is used to analyse a new real traffic it will suffer from high false alarms and low detection accuracy. These phenomena are usually caused by the changes in network patterns, which would lead to an early phasing out of such a model and a triggering of model regeneration or updating phase. This could be linked to the inefficiency of using a fixed discriminating threshold for such ID models. For example, a network under high volume attacks, such as Denial of Service (DoS) or scan attacks, would have different class (normal to attack) distributions than when it is under low volume but stealthy attacks such as SQL injection and Command-and-Control (C&C).

Therefore, this thesis is intended to address this problem by investigating the effect of adapting the discriminating threshold (specifically to the evaluated network traffic) on the accuracy (i.e. the Geometric Mean of Accuracy) of such models and compare the results with the use of a fixed threshold. This investigation will be done by comparing such effects on traffic collected

at different times with existing variability. Further, the ability of different types of ML algorithms to adapt to traffic changes will be analysed.

1.2 Motivation

The key impact of ever evolving LAN network traffic is linked to the difficulty of building ID models that can address all possible variations in traffic patterns. This in turn is linked partially to the challenge imposed in collecting training data for every imaginable scenario. Traditionally, the model development process would use a training data to build a detection model which can then be used to predict future data. However, as traffic evolves over time, the learned model would usually experience a large decline in detection rate. As a result, new data would be needed to train another model.

As will be illustrated in more detail in the coming chapters; most of the learning and classification methods are based on a number of key assumptions [8, 9], such as: (i) the equal representation of classes, (ii) the equal representation of sub-concepts for a specific class, (iii) the similar class-conditional distributions of all classes, and (iv) the pre-defining and knowledge of all the values of the attributes for all records in the dataset. With traffic evolution the reality is that most, if not all, of these assumptions are violated in real environments, as new traffic will start to exhibit different statistical properties to those of the training data.

Traffic evolution can introduce unpredictable differences between the training data and the testing data. These differences can take various forms; for example, class distributions might differ in the new data than those used to build the ID model, and even new classes might emerge over time. In addition, class balance (also known as data balance) can play an important role on the accuracy of constructed models, which could be affected as a result of pattern changes. Traffic variability may also bring about differences in feature importance. These effects (collectively or individually) might render the learnt model outdated sooner than anticipated.

However, the current methods to deal with these effects (in a batch-based setup) will attempt to generate a new model, which may consume additional resources in collecting and labelling new data to be used to learn that new model.

Many studies have attempted to address some of these issues in real-time setups by tuning the detection parameters of the ID models, while others have introduced ensemble methods for data stream setups. However, there is insufficient empirical work to analyse the threshold adaptation of model predictions in binary batch-learning (*offline learning*) setups, which refers to the process of building an ID model using the entire training data (full-batch learning); or when the model is updated or rebuilt after certain update criterion is met (mini-batch learning), such as the lapse of a specific time period or after a number of training instances have been collected [10].

The low detection accuracy of such score-based anomaly ID models, in batch-learning setup, could be linked to the use of a fixed discriminating threshold, which in turn could result in an inaccurate reading of the accuracy that is far lower than their actual optimal accuracy. This might explain the early termination of such ID models. As a result, adapting the discriminating threshold to the predictions of the evaluated network traffic would provide an accurate reading of the actual accuracy of the ID model. Understanding this may lead to an improvement in detection accuracy and hence an extension in the lifespan of the ID models.

1.3 Scope of the Research

The main objective of this thesis is to improve the accuracy (i.e. the Geometric Mean of Accuracy) of a score-based anomaly Intrusion Detection (ID) model. To this end, the key approach undertaken by this research to fulfil this objective is by tuning the discriminating threshold specifically to the predictions of the network traffic evaluated by such a model.

In other words, it is intended to evaluate the potential of prediction threshold adaptation to improve the accuracy (i.e. the Geometric Mean of Accuracy) of binary anomaly ID models developed using various ML algorithms for batch-based tasks. This study focuses on detecting anomalies (intrusions) in Local Area Network (LAN) traffic, where the payload (content) of such traffic is out of the scope of this thesis. The traffic of such networks are assumed to be fully captured with no (or negligible) packet loss, at a strategic location with full visibility of exchanged communications of the monitored (protected) information systems.

In this thesis no IDS evaluation measures other than the accuracy of the system is addressed. To explain further, as this thesis is focussed on batch-based ID models, the *timeliness* measure, which measures the total delay between the start time of the attack and the response time of the system [11], is out of the scope. Similarly, as binary ID models form the core subject of this study, there will be no focus on the *completeness* measure, which evaluates the coverage of the intrusion space, that is to assess if an IDS (detection model) can detect all or most of the attacks [11]. Moreover, as this thesis aimed to examine each detection model independently it does not address the *interoperability* measure, which assesses the capability of an IDS to correlate information from multiple sources [12] or to interoperate with other IDS in a multi-IDS environment [13].

System scalability and resources utilisation (CPU and memory) are also out of the interest of this thesis. These issues are considered as engineering problems that can be the focus of future studies.

The following list provides an explanation or description of some of the key terms used throughout this thesis:

- A ***training*** dataset is the data used to learn or build a ID model.

- An *evaluation* or *test* dataset refers to the data that are assessed by the detection (classification) models or used to evaluate these models.
- The *validation* data describes the data used to set or fine tune the parameters of the ID models.

This work assumes existing variability between the network traffic used as the training data and the test data, where no specific time interval has been defined for such variability to occur. However, measuring the degree of variability (drift) is out of the scope of this study. Moreover, this research assumes that the labels of the validation data will be available whenever they are needed, and that the source of these labels is not pre-specified. Hence, these labels can come from another IDS or security analyst.

1.4 Research Hypothesis and Questions

The following core research hypothesis has been evaluated and tested in this thesis:

“In a binary batch-learning setup, prediction accuracy of a score-based anomaly intrusion detection model can be improved by adapting the discriminating threshold specifically for the predictions of the evaluated network traffic.”

Examining this hypothesis requires the investigation process to address various levels of statistical differences between the training data and the test data, in order to consider diverse model development scenarios that echo real world practices, and to analyse the feasibility of the threshold adaptability approach based on a small sampled subset of the overall data. To investigate the hypothesis above, the following questions have been addressed:

Q1. How will the detection accuracy of an adaptive discriminating threshold of the predictions of a batch binary-based anomaly ID model compare to the accuracy of a fixed threshold?

Conventional methods will set a prediction threshold for a detection model only once where this threshold will be used to classify future data regardless of any new patterns (drifts). This question is investigated empirically (in **Chapter 4**) by comparing model detection accuracies using fixed and adaptive thresholds. These models are trained and evaluated with datasets, which have controlled degrees of similarities and differences between their statistical properties (concept drifts).

Q2. Can the adaptation of the discriminating threshold improve the accuracy of a binary-based anomaly ID model when evaluated network traffic has different salient features than those used to build the predictive model?

Currently, model development might be performed after some analysis tasks, such as feature selection or data balancing. The aim of these tasks is to improve model detection, however, most of these tasks have to be performed on the training data available at the time. When such models are then used to evaluate new data, their prediction performance (accuracy) may diminish as a result of statistical differences of such data, which could have different sets of important features to those used to build the model. **Chapter 6** introduces an empirical study to address this question and investigates the effect of threshold adaptation in such scenarios, when a newly generated dataset has been used (as outlined in **Chapter 5**) for these analyses.

Q3. Can the optimal discriminating threshold be identified using a labelled small sample of the evaluated network traffic under the batch-learning setup?

Threshold adaptation requires knowledge of the true labels of the evaluated dataset to optimise for some performance measure (i.e. Geometric Mean of Accuracy). However, labelling all of the evaluation data (network traffic) in a domain like network security would be a nearly impossible task. **Chapter 7** outlines an experiment undertaken to study threshold adaptation using a small subset that is sampled from the dataset being evaluated. To answer this question, the effects of different sample sizes and sampling strategies on the adaptation process were analysed, in addition to exploring which of the ML algorithm's models were more suitable for the adaptation process. It also investigated the effect of different error rates (of the sample's true labels) on the threshold tuning process.

1.5 Research Approach

The discussion in this thesis describes an empirical study undertaken to address the research hypothesis set out above and its related research questions. It provides evidence on the usefulness of the proposed approach on various datasets (synthetic and domain specific) using different ML algorithms (C5.0, Random Forest and Support Vector Machine). A factorial research design was employed to meet the aim of this study. Further details of the decisions made during this research are provided in **Chapter 3**.

1.6 Contributions

The following list highlights the key novel contributions of this thesis to the current state of knowledge in this field:

- **Scientific contributions:**

- C1. A new evaluation method was employed to assess the performance of models, by performing **prospective sampling** (as discussed in **Chapter 2** and, applied in **Chapter 4** and **Chapter 6**) instead of the conventional K-folds Cross-Validation method. This new evaluation technique mimics real world setups.

- C2. A proof of concept analysis was conducted using different ML algorithms (C5.0, Random Forest and Support Vector Machine) with synthetic datasets to compare the capability of models to predict test data that had different statistical properties to the training data. The analysis revealed that the Random Forest (RF) algorithm was the most accurate and the most adaptable of the three algorithms (**Chapter 4**).
- C3. A thorough analysis of the performance (detection accuracy) of the three ML algorithms was conducted, and the Random Forest (RF) algorithm was the best at classifying new traffic in the **STA2018** dataset (**Chapter 6**). This analysis included the evaluation of different models (with different sets of features and data balances) for every ML algorithm.
- C4. An analysis of the performance of the different ML algorithms on different network traffic (with changing behaviour) revealed that the performance of these ID models did not reach their optimal capacity if their predictions were not adapted for the newly evaluated traffic (**Chapter 6**).
- C5. An investigation was conducted to analyse adapting the discriminating threshold of model predictions using a (small) random sample of the evaluation traffic (**Chapter 7**). This investigation revealed that this approach was able to correctly adapting a model's predictions with more than 95% of their optimal accuracy when the true labels of only 10% of the original data were used for this correction (threshold tuning) task. Different sample sizes were tested and the predictions of the Random Forest (RF) models were able to use a sample as small as 0.05%, of the evaluated dataset, to compute an adaptive threshold with up to 90% of the overall accuracy of the optimal threshold.

C6. A deeper analysis was conducted to investigate the performance (detection accuracy) differences between various ML algorithms in order to investigate the causes of differences in their model detection capabilities (**Chapter 7**). Most existing studies compare the performance of different ML algorithms but they do not extend their analysis to the level of prediction results returned by these models. This study extended its analysis to investigate why such differences could occur, and as a result, revealed a possible relationship between the *number of unique prediction scores* and their *ranges* (returned by ID models) in the overall performance of these model.

- **Practical contribution:**

C7. An evaluation of the generation process of the KDD 1999 dataset, which revealed the faulty nature of this dataset. In addition, the evaluation demonstrated a novel way of linking connections from KDD 1999 to their originals in the DARPA 1998 dataset (**Paper is accepted for publication [1]**).

C8. A new dataset was generated (**STA2018**) by transforming the network traces of the UNB ISCX Intrusion Detection Evaluation DataSet 2012 [14] into a suitable format for ML and DM tasks (**Chapter 5**). The generation process used traffic trace files to extract 193 basic features, which were then expanded to 550 features by employing Onut's feature classification schema [15]. Every record in the resultant dataset profiled an independent connection, making it suitable for ML algorithms. This dataset will be publicly available.

C9. A number of faults in the labelled flow files of the UNB ISCX Intrusion Detection Evaluation DataSet 2012 were identified and outlined, and these faults have been communicated to the authors of this dataset (**Chapter 5**).

C10. Basic feature extraction was scripted using Bro [16] software (**Chapter 5**). In addition, related parts in Onut's feature classification schema [15] were implemented from scratch using Java. The full source code will be publicly available for researchers.

C11. An extension was made to the visual illustration of the Critical Difference Plots, first proposed by Demšar [17] (**Chapter 7**). The extended version (Multi-CD) combines multiple Critical Difference Plots into a single figure, so that any changes in the ranking of various factors under different effects are more pronounced.

1.7 Research Output

Al Tobi, Amjad, and Duncan, Ishbel. KDD 1999 Generation Faults: A Review and Analysis. *Journal of Cyber Security Technology*, 2018. [1].

1.8 Thesis Structure

The remainder of this thesis is structured as follows:

Chapter 2: presents the background to this field of research and introduces related topics discussed within the thesis. It also highlights the key problems and solutions in this area and the latest work related to these issues.

Chapter 3: provides an overview of every experiment conducted in the thesis, and an introductory background of every ML algorithm used in the experimental evidence. It also discusses *the analysis methods, the adopted research design and the chosen statistical tests* that are used in all experiments of this thesis.

Chapter 4: establishes a Proof of Concept (PoC) for the core focus of this thesis. It details the experiments conducted to address the first research question, which investigated the feasibility of threshold adaptation for model predictions of evaluation datasets with

various degrees of statistical similarities to the training data used to learn the ID model.

Chapter 5: provides a detailed discussion on the generation of the *STA2018* dataset. It describes the preparation of the data and the process of transforming raw network traffic from the *UNB ISCX 2012* [14] dataset into a suitable format for ML algorithms. The resultant dataset (*STA2018*) contains labelled records for seven days of simulation traffic, where each connection (session) is profiled using 550 features. The *STA2018* dataset was subsequently used in later experiments.

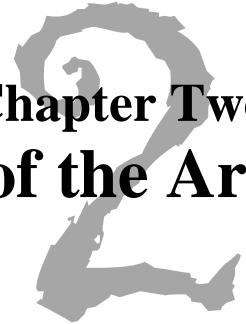
Chapter 6: presents the experiments undertaken to investigate the second research question. They aimed to examine the effect of threshold adaptation (tuning), specifically to the evaluated data, on the overall performance of the models and compare the performance to the use of a fixed threshold. This analysis included various model development setups that aimed to emulate real-life practices.

Chapter 7: addresses the third research question by exploring the potential of selecting the optimal threshold for the whole evaluation data by using a randomly sampled subset. The true labels of the small subset were used to set the prediction threshold for the entire test data. This chapter also analyses and compares the effect of various factors, such as *sample size*, *sampling strategies* and *labelling error rates*, on the optimal threshold selection for each of the analysed ML algorithms under different model setups.

Chapter 8: concludes with a summary of the main results and findings from this research, and suggests some potential areas for future studies.

Chapter Two

Literature Review and State of the Art



Computer security was raised as a concern nearly fifty years ago by James P. Anderson in 1972 [18, 19]. Anderson stressed the urgent need for research and development in this field to ensure secure information systems. In 1980, early research led to the introduction of the concept of ‘Intrusion Detection’ (ID) [20]. The term *intrusion* is used to describe any illegal endeavour to gain access to, or to manipulate, information, or any attempt to sabotage a system by making it inaccessible, unusable or unreliable.

In a landmark piece of work to automate the detection process, Denning [11] proposed the first ID model which formed the basis for all subsequent advancements in the field. This chapter introduces ID and discusses the present state of affairs in this area of research.

2.1 Intrusion Detection (ID)

Intrusion Detection Systems (IDS) are important in the security paradigm of any information system to protect systems from internal and external threats [21]. Their aim is to distinguish between legitimate and anomalous actions [22, 23]. As a result, many different types of IDS exist, including commercial and open source systems. IDS can be classified either by their physical positioning or by their detection methods, both of which are explored in more detail in the following discussion.

2.1.1 Types of Intrusion Detection

The physical positioning of different IDS defines their role in the overall security structure of an organisation. There are two main categories of IDS based on their position: *host-based* and *network-based* IDS.

2.1.1.1 Host-based IDS (HIDS)

Host-based IDS (HIDS) are used to monitor activities at the host level [22, 24, 25]. In such systems, all the network traffic, resource utilisation and audit trails of a specific device are monitored to detect any suspicious activities or attack attempts [26]. Such systems monitor the internal activities of a system (CPU, RAM, processes, files, *etc.*) [27]. However, they could induce an overhead on the performance of the monitored host due to their consumption of resources [21]. In addition, they require extra resourcing and effort to be deployed on every host [21]. For instance, in a Cloud environment, this deployment could extend to include every Virtual Machine (VM) running in every hypervisor [28].

2.1.1.2 Network-based IDS (NIDS)

With the increased use of network services, different types of intrusions and attacks have started to emerge, resulting in a growing need for Network-based IDS (NIDS). These systems are usually deployed in a strategic position [21] where they can monitor and analyse the traffic exchanged to protect multiple systems. NIDS are faced with a high volume and a wide diversity of traffic. This could impose greater complexity in detecting intrusions as trade-off to their ease of implementation and deployment [26, 29]. Encrypted traffic adds another layer of complexity to these systems as in many such cases the traffic needs to be decrypted before being analysed [28]. Furthermore, in a Cloud environment, the NIDS could experience serious limitations when the virtual network inside a hypervisor comes under attack, i.e. if the NIDS has no visibility over that virtual network [28], as a result of its deployment outside the hypervisor. Nikolai and Wang [30] have proposed a *Hypervisor-based Cloud IDS* to address such limitations.

Every advance in technology, such as Clouds and the Internet-of-Things (IoT), adds more challenges to the list of problems that need to be addressed. The recent growing interest in detecting attacks on wireless networks is driven by their widespread deployment as well as their vulnerability to attacks compared to wired networks [25].

2.1.2 Methods of Intrusion Detection

The detection mechanism which defines the core functionality of IDS can be categorised into two primary methods: *misuse-based* and *anomaly-based*.

2.1.2.1 Misuse-based Detection

Misuse-based IDS (such as Signature-based or Knowledge-based IDS) match known dubious patterns, which are termed as definition of attacks [25, 31], with actual observed activities, such as recorded behaviours in audit trails or network traffic. This matching process is usually accomplished by translating known system vulnerabilities [23] and intrusive patterns into some form of signatures. Although, this approach is effective in detecting known attacks, it fails to detect new (zero-day) intrusions [27] or even variants of known attacks [23, 31-34], such as polymorphic worms [26, 35]. Many researchers have proposed solutions in attempts to address this issue, such as *alert verification* [36, 37], *proactive approach* [38] and *ensemble of multiple classifiers* [39].

Another challenge facing this approach is signature definition as it is difficult to define signatures that cover every system vulnerability [23] and all variations of a possible attack [32, 33]. These systems consume time in maintaining the knowledge base due to the required effort in keeping patterns or signatures up to date [25]. Despite the many limitations of these types of IDS, most of the existing solutions (both commercial and open source) such as Snort¹ and Suricata² fall into this category. Various techniques are used in misuse detection methods, such

¹ <https://www.snort.org/>

² <https://suricata-ids.org/>

as *pattern recognition, rule-based expert system, Data Mining, etc* [31]. The use of DM techniques can be controversial as many researchers apply them in a misuse context, while others use them for anomaly detection.

Many of the misuse methods, such as pattern matching for signature-based IDS, require high computational resources where their utilization to process the vast amounts of network traffic forms a bottleneck. Therefore, techniques such as those proposed by Bellekens *et al.* [40] which use the General Purpose Graphics Processing Unit (GPGPU), could provide a promising solution to address this problem for this kind of IDS due to the efficient computational processing abilities of the GPGPU.

2.1.2.2 Anomaly Detection

The core functionality of anomaly-based (or behaviour-based) IDS is based on building a profile of the normal activities for the system [25, 28, 31]. The main problem with this approach is the concept of ‘normality’ which could provide an appropriate solution if it can be defined accurately. This approach assumes that all intrusive (anomalous) activities must deviate from a normal pattern [21, 23, 28, 32, 33, 41]. It also assumes that anomalous activities are rare in comparison to normal ones. As a result, many researchers have attempted to solve this problem by introducing formal models that express the relationships between the core parameters involved in the system dynamics [12] to build a formal model of normality. Every activity evaluated with such a model is then classified as an anomaly if it deviates from the normal pattern [21, 31].

However, anomaly-based techniques suffer from a large number of false alarms [23] because of their inability to adapt to constantly changing events [25]. Also, due to the underlying assumption of the rarity of the anomalous activities, many of these techniques become erroneous (high false alarm rates) when anomalies hit the system in large quantities [12]. This weakness calls into question and challenges the applicability of these techniques in a production

setting and could explain the dominance of signature-based IDS in such environments. Also, as Modi *et al.* [28] have pointed out, many of these techniques would require a longer time to identify intrusive activities. As such, Buczak and Guven [27] consider misuse-based detection methods to be ‘proactive’ due to their continuous checking of activity signatures against known attack patterns, while anomaly-based techniques are considered ‘reactive’.

There has been a growing interest in anomaly-based techniques due to advancements in this field, the continuous enhancement of physical resources, and the ability of these methods to detect novel (zero-day) intrusions [21, 23, 25, 31]. Any novel attacks that are detected can be used to develop signatures for the misuse-based IDS [27], if required. Some of the main techniques used in anomaly-based methods are discussed in more detail below.

Various approaches have been studied and investigated to address anomaly-based detection. In this area, the most commonly used techniques are *statistical*, *cluster-based* and *classification-based* methods [31]. Many of the anomaly-based methods are applied to different placement types, such as host-based or network-based types, but their aim is to detect any deviation from the norm. Various ML and DM techniques are used for these tasks. Although, many of these ML and DM techniques are used in both misuse and anomaly detection methods, the usage method defines their context. In misuse-based detection, ML and DM techniques are used to generate signatures and patterns of attacks that can be used later to detect these attacks, whereas anomaly-based detection uses these techniques to build models to define the normal behaviour. These models are then used to detect any activities that deviate from the norm and flag them as anomalies. Some of the key methods used in these techniques are outlined in more detail below.

Many of **statistical methods** used to analyse traffic are based on the theory of abrupt changes [12]. These methods basically work by measuring the means and standard deviations of certain variables to flag anomalous behaviours when they exceed predefined thresholds or

probabilities [25]. They monitor activities and profile them statistically based on measuring specific variables for an extended period of time [31]. After that, any sudden or unexpected events, which cause significant deviation, are reported [42]. Such models are more useful to detect threats of high volume, i.e. DoS and probing attacks, where more stealthy attacks will have a high probability to evade their net. As a result, Staniford *et al.* [43] proposed the Stealthy Port scan and Intrusion Correlation Engine (SPICE) which uses statistical techniques to detect stealthy port scans by applying a frequency-based mechanism that assigns higher anomaly scores to those packets that are observed less frequently. Many other methods are discussed in the literature such as Hierarchical Intrusion Detection (HIDE) [44]; Packet Header Anomaly Detection (PHAD) [45]; PAYL [46]; LERAD [47]; and Flow-based Statistical Aggregation Scheme (FSAS) [48]. One of the main drawbacks of these methods is that they can also be trained by an adversary.

Clustering-based methods aim to group instances (network connections or activities) into collections called *clusters* [12]. These methods are based on the assumption that similar instances should be close to each other and apart otherwise [27]. With these methods, no prior knowledge is required of the labels or classes (unsupervised) [24], and normal instances are assumed to be the larger cluster [23]. Bhuyan *et al.* [12] suggest that these techniques are best performed at the exploration stage of the DM process. Many clustering methods are discussed in the context of IDS, such as hierarchical clustering [49] k-means [50], Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [51], Simple Logfile Clustering Tool (SLCT) [52] and many more. Clustering methods do not require training which saves effort for a system administrator in collecting and labelling data [27]. However, data dimensionality may impose serious limitations on these techniques at high dimensions, because most of them are based on using distances between data points as an evaluation measure, a measure which loses accuracy as the number of dimensions increases [31]. Furthermore, at higher dimensions, many

features will not be meaningful for certain clusters which means cluster-based techniques suffer from the *local feature relevance* problem. This means that global feature filtering or selection will not be satisfactory as different clusters could form in different subspaces [53].

Classification-based techniques are used to assign instances into one of a number of predefined categories (or classes) [23, 26]. These techniques can perform binary (normal or attack) or multi-class classification [31]. Many of these techniques have been explored in detecting anomalous network traffic using K-Nearest Neighbour (KNN) [54], Decision Trees (DT) [55], Support Vector Machine (SVM) [56], Neural Network (NN) [57] and Bayes Classifier [58] amongst other techniques. Classification methods tend not to suffer at higher dimensions as their outcomes show more stable results than cluster-based techniques. However, unlike cluster-based methods, they require labelled training data to build their prediction models [12]. In addition, class distribution (balance) in training data could impose a challenge on the learning capabilities of these techniques [12], an issue which is discussed further in **Section 2.2.3.2**. Moreover, they tend to mainly suffer from their high consumption of resources [12]. A number of studies have proposed various solutions to address this problem, such as selecting the best subset of salient features (see **Section 2.2.3.1**) which would result in a large improvement in resource utilization [29]. Bhuyan *et al.* [12] have argued that classification-based techniques cannot predict novel attacks until retraining with the new attacks has been performed. This assumption is based on granular level classification, where every type of anomaly is categorised into one class. More holistic forms of classification, such as binary classification, could be able to predict new types of attacks under a more general attack class. Some of the well-known systems that employ classification principles include Repeated Incremental Pruning to Produce Error Reduction (RIPPER) [59], Automated Data Analysis and Mining (ADAM) [60] and Dynamically Growing Self-Organizing Tree (DGSOT) [61].

According to Bhuyan *et al.* [12], the research community are more focused on and most interested in classification-based techniques. Their popularity stems from their flexibility in training and testing process, as well as their high detection rates for known attacks conditioned on the appropriate setting of their prediction threshold. Although this fact, i.e. appropriate threshold setting, is known in the field, to the best of the obtained knowledge, no research has been conducted to analyse this in greater depth and to study the effect of threshold adaptation under different parameters.

2.1.3 Hybrid and specialised approaches

A number of hybrid systems have been proposed to address the limitations of each ID type or method individually by combining multiple techniques [31] in order to achieve more extensive and more accurate detection [25], and to detect the known and unknown threats [12]. As a result, many systems have tried to exploit the advantages and strengths of various methods by: combining misuse and anomaly detection [62-68]; combining multiple anomaly-based detectors [69-71]; or even combining all approaches together (misuse, anomaly, host-based and network-based) [72]. Although these hybrid approaches help in building better IDS, unfortunately they tend to suffer from a high computational cost [28]. However, with advances in technology, they are starting to dominate the research field [27].

The new technological advancements, such as Big Data and the Internet of Things (IoT), have brought new challenges with them due to the explosion of information. Many proposals have been put forward to address the security concerns surrounding these problems, such as Collaborative Intrusion Detection Systems (CIDS) {*centralized* [73-76], *hierarchical* [44, 77-79], *distributed* [80-87]}, [21], Wireless-based IDS (WIDS) [25] and Cloud-based IDS {Hypervisor-based IDS} [88], *etc.*

Due to the diversity of network traffic as a result of so many services, more specialised IDS have started to emerge, such as application-based IDS (i.e. email, FTP, HTTP) [89, 90], protocol-based IDS (i.e. TCP) [91, 92], and content or payload-based IDS [93, 94].

As technology advances, more sophisticated IDS are being proposed which combine approaches from multiple domains to address specific technological limitations. Such sophisticated approaches could cause difficulties in classifying solutions into a specific class or category. Therefore, other approaches and techniques are discussed in the literature on how to address the ID problem. They include, but are not limited to, Ensemble-based [95, 96], Fusion-based [97], Big Data [98], Genetic Algorithms (GA) [99-104], Self-Organizing Map (SOM) [105-107], Artificial Neural Networks (ANN) [108-112], Artificial Immune Systems (AIS) [113-116], Fuzzy sets [117-124], Rough sets [125-129], Ant Colony [130-132], and ontology and logic-based [133, 134] approaches.

2.1.4 Batch versus Ensemble Intrusion Detection

All of the approaches and methods discussed above were initially proposed to produce pattern signatures or prediction models in a batch-learning setting, where a finite dataset was used in the learning task [135]. However, with the explosion of data, more challenges have emerged that have in turn introduced new fields and trends in handling Big Data, which are known as **ensemble** or **data stream** methods. Traditional (batch-learning) methods of using learning algorithms to develop predictive models has been one of the areas affected due to their inability to handle many challenges. Although data stream is not the focus of this thesis, it has been introduced here to illustrate the new trends in the area and to introduce some of the key terms in the field, given that some of these terms also affect conventional (batch-learning) methods of developing prediction model.

In their seminal review, Gomes *et al.* [135] listed a number of the challenges faced by the conventional (batch-learning) ML and DM techniques. Amongst many of these challenges,

which data stream methods aim to resolve, are the enormous volumes of data that need to be processed, limited time and memory resources, the emergence of new classes over time, and temporal dependencies [136]. However, the most significant problems which have a direct effect on predictive models are *concept drift* and *feature drift*.

Concept drift [137] is a term used to describe the situation in which data distribution varies over time [138]. It has been categorised by Aggarwal [139] into four groups: *abrupt*, *incremental*, *gradual*, or *recurring*. Concept drift describes the nature or phenomena of network traffic and is often neglected in studies which investigate anomaly-based network IDS in a conventional (batch-learning) setup.

The term **Feature drift** refers to the relevance of features over time [140, 141] as changes in data patterns will incur varying levels of relevance of features. In other words, when a group of features becomes irrelevant (or relevant) to the learning process at a certain point, then feature drift has occurred [142].

A lot of research has emerged to look at ensemble learning for data streams to address these issues, in addition to the proposal of many methods, techniques and frameworks, such as, *Massive Online Analysis (MOA)*³ [143]. Such lines of research are based on the assumption that building a strong prediction model is a challenging task and, therefore, developing multiple weak models is feasible in order to boost the models by strategically training and uniting them to create a strong model [144-147]. In these ensemble or data stream techniques when a concept or feature drift is detected, new models are built and the least performing models will be discarded.

³ <https://moa.cms.waikato.ac.nz/>

2.2 Datasets

Datasets provide important means to conduct empirical evaluations and to undertake comparative analyses of various methods and techniques. However, the network security domain, especially the network ID field, lacks good quality datasets to analyse and test new techniques and algorithms. Many factors contributing to this, including the fast evolution of this field [31]. The highly evolving and changing nature of network traffic and the vast amount of different attack types introduced on a daily basis, have made any attempt to keep producing up to date, good quality data a daunting task. As a result, many recent studies use datasets that are outdated and as old as KDD 1999. Another factor is related to privacy concerns; while many real and up to date datasets have been used in various studies they have not been made publicly available because of such concerns. Furthermore, to the best of the author's knowledge, there are no datasets that simulate intrusions with new technologies such as a cloud environment or the IoT. Therefore, finding a recent study similar to the one undertaken by Deng *et al.* [148] (who discussed security issues and ID within IoT applications by conducting analyses based on the KDD 1999 dataset), would raise concerns about its findings.

While there are many publicly available datasets, most of them are over a decade old although they are still being used in the most recent studies. In general, in addition to the lack of datasets in the ID domain, the existing datasets have many limitations, such as lack of wide range and up-to-date attacks. The sections below list the most widely known datasets and benchmarks.

2.2.1 Raw datasets

Many of the widely known datasets used in network ID studies contain raw data. These data usually consist of pure network traffic and traces between hosts and services. Some of them contain the host's audit files which are also in a raw format. The ML and DM techniques require this data to be pre-processed in order to extract the required features and attributes before they can be used to produce predictive models.

DARPA datasets⁴ were the first of their kind to be publicly available for researchers. They were generated as part of an evaluation task of various Intrusion Detection Systems. They were a result of a joint project between the Air Force Research Laboratory (AFRL/SNHS), Defence Advanced Research Projects Agency (DARPA ITO) and the MIT Lincoln Laboratory. This project generated three sets of data: *DARPA 1998*, *DARPA 1999* and *DARPA 2000*. However, the most widely used dataset in the literature is DARPA 1998. These datasets include raw data from host audits and network traffic traces.

The **UNIBS-2009 dataset**⁵ consists of three days of network traces from twenty workstations. These traces were collected on the edge router of the campus network at the University of Brescia in Italy. This traffic contains multiple services and protocols, such as *Web*, *Mail*, *Skype*, *Peer-to-Peer applications (BitTorrent and Edonkey)* and many other protocols. For this dataset, all payloads were stripped off and all of the addresses were anonymised.

UNB datasets⁶ were generated by a leading team in this field. There are multiple datasets available at the time of writing this thesis, such as *ISCX IDS 2012*, *CIC IDS 2017*, *ISCX VPN-nonVPN traffic* and the *ISCX Botnet datasets*. Both the ISCX IDS 2012 and the CIC IDS 2017 datasets consist of seven days of network traffic captures, where the traces contain full captures and payloads, and are not anonymised. They contain a variety of traffic using up-to-date services with recent types of attack scenarios.

Other datasets can be found from various sources such as DEFCON⁷ [149], CAIDA⁸ [150] and LBNL⁹ [151]. However, these datasets have a number of limitations in comparison to the ones discussed above. For example, the captures of DEFCON and CAIDA are very small and consist

⁴ <https://www.ll.mit.edu/ideval/data/index.html>

⁵ <http://netweb.ing.unibs.it/~ntw/tools/traces/>

⁶ <http://www.unb.ca/cic/datasets/index.html>

⁷ <https://www.defcon.org/>

⁸ <http://www.caida.org/>

⁹ <https://www.icir.org/enterprise-tracing/Overview.html>

of traffic for short periods. They are targeted on a specific scenario and contain particular attacks [12]. The LBNL dataset is limited only to header traces and all packets are anonymised, which could result in a distortion of the topological structure of the network. TUIDS¹⁰ is another dataset [152-154] that is discussed in a few publications by its authors.

2.2.2 *Processed (pre-formatted) datasets*

As most of the ML and DM algorithms and techniques cannot handle raw data, it is necessary to convert them into a suitable format. Some of the datasets outlined in the literature are used for that purpose and many of them are much used in several studies.

The **KDD Cup 1999**¹¹ [155, 156] is the most famous and most commonly used dataset in the domain of network ID. This dataset is a transformation of network traces from the DARPA 1998 dataset. The transformation aimed to process the raw data into a format suitable for ML and DM tasks so that every connection in the network traces was profiled with 41 features. The transformed dataset consists of multiple attack types that are categorised into four classes: *probing*; *DoS*; *remote to local (R2L)*; and *user to root (U2R)*. All the records in this dataset have no host addresses and their chronological order is distorted as their start times were stripped off. Regardless of these limitations and the many criticisms discussed in various studies, this dataset has been used in recent studies, up to and including 2018. However, Tavallae *et al.* [157] have analysed this dataset and investigated its poor performance, and they proposed some fixes by removing redundant records, resulting in a variant called **NSL-KDD**. A thorough analysis and investigation of the **KDD Cup 1999** dataset can be found in Al Tobi and Duncan [1], where far more serious limitations, other than redundant records, have been identified.

¹⁰ TUIDS is not accessible as the authors do not respond to access requests and all links, provided in their publications to the dataset, are broken.

¹¹ <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>

In 2006, the **Kyoto datasets**¹² were generated by extracting 24 statistical features from the raw traffic of a honeypot system deployed at Kyoto University in Japan [158]. Extracted features include non-anonymised host addresses. However, as this dataset only contains traffic that was directed to the honeypots, this restricts the view of that network's traffic [159]. All of its normal traffic was simulated by generating DNS and mail traffic, which does not make it a realistic representation of real world traffic.

The **GureKddcup**¹³ [160-162] dataset was put forward in 2008 by Perona *et al.* [161]. It is a retransformation of the DARPA 1998 dataset and is similar to the KDD Cup 1999 dataset. However, this transformation is much cleaner as it avoided many of the limitations of the KDD Cup 1999 dataset (such as redundant records, the availability of host addresses and connection timestamps). Although this dataset has existed for a long time, it is not as widely used in studies as the KDD Cup 1999.

The **Sperotto's (Twente) dataset** was generated in 2009 by Sperotto *et al.* [163] by capturing traffic through a honeypot deployed in the network of the University of Twente. The honeypot provided three services (OpenSSH, Apache web server and Proftpd) and was directly connected to the internet. Although 98% of its flows are labelled, it contains limited attacks compared to other available datasets.

The **UNSW-NB15 dataset**¹⁴ was created by Moustafa and Slay [164, 165] using the IXIA PerfectStorm tool to generate a mixture (benign and anomalous) of modern network traffic. This dataset contains nine different attack types: *Fuzzers*; *Analysis*; *Backdoors*; *DoS*; *Exploits*; *Generic*; *Reconnaissance*; *Shellcode*; and *Worms*. It contains a total of 31 hours of simulation

¹² http://www.takakura.com/Kyoto_data/

¹³ <http://www.sc.ehu.es/acwaldap/gureKddcup/>

¹⁴ <https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-Datasets/>

time (16 hours in the first day and 15 hours in the second day) with a total of 1,140,044 connection records, with each record profiled using 49 features.

While there are many other datasets, KDD Cup 1999 and NSL-KDD are the most commonly used in the literature. However, as pointed out by Catania and Garino [31], many studies have used their own datasets which have not subsequently been made publicly available. This was confirmed by Abt and Baier [166] who showed in their study that only 10% of the papers they surveyed had released their datasets. Moreover, in many cases, no detailed information had been provided on the process of generating these datasets, which undermines a key principle of scientific research, the replicability of experiments [31]. Despite this, many frameworks for proper and replicable dataset generation have been proposed [14, 166-168].

2.2.3 Common issues and pre-processing tasks

Prior to any training phase, datasets could undergo some pre-processing phases to address issues related to the data itself or the model generation stage. Such pre-processing includes data cleaning which involves dealing with missing, noisy and incomplete values. It might also include data transformation, such as normalisation and standardisation. The main pre-processing tasks which have a direct influence on model performance are *feature selection* and *data (class) balancing*. Each of these issues is reviewed next.

2.2.3.1 Feature selection

Feature selection is one of the most important tasks in the data analytics process. It aims to reduce model complexity and generation time by selecting salient features that best capture patterns within the data. It also removes redundant and irrelevant features that could reduce the generalisation capability of a model and thus avoid the problem of overfitting [169, 170]. Avoiding the curse of dimensionality [171] (where the number of features are greater than the number of instances in the dataset) is another important reason for performing a feature

selection process as some algorithms are particularly susceptible to this problem, causing the learned model to overfit [171].

It is worth noting that feature selection is different from dimension reduction. In feature selection the best subset of features is selected, while dimension reduction creates a new combination of features by projecting the original feature space into a new low dimensional feature space. Principal Component Analysis (PCA) and Singular Value Decomposition (SVD) [172-174] are the most common dimension reduction techniques. However, dimension reduction methods are out of scope of this thesis due to their limitations. For example, these methods make it difficult to determine the level of influence of any individual feature. These methods also require the data to be pre-processed (i.e. normalised) as unpredictable issues might arise when data with different scales is used.

Feature selection algorithms generally fall into one of three categories: wrapper, filter, or embedded methods [172]. **Wrapper** methods [173-176] evaluate a subset (group) of features using a predictive model, which is assessed using hold-out (discussed in **Section 2.3.1**) data that were not used in the training phase of the model. Based on the error rates of these models, the feature subsets (groups) are assigned a score. As the wrapper method is considered a search problem, different search methods are applied throughout the selection process, such as, best-first [177, 178], random hill-climbing [177, 179], and forward and backward passes [180-182]. These methods suffer from intensive computations as a result of fitting models for every evaluated subset of features.

Filter methods [173, 174, 183, 184] use statistical measures, known to be fast to compute, as their evaluation criteria rather than the model's performance i.e. accuracy. These measures usually attempt to assess how useful a feature might be. However, as this assessment is not linked to the model's performance, the selection process is not usually adjusted for a specific type of model. This method can lead to the selection of more general features which, in turn,

can result in the generation of models with lower powers of prediction. There are many measures used in this group such as Relief-based Algorithms [185-187], Mutual Information [188, 189], minimum-Redundancy-Maximum-Relevance (mRMR) [190], Chi squared test [191-193], Information Gain, Correlation Coefficient scores and many others. Following the use of these methods, scores are assigned to every feature. The selection or elimination of features using this method is based on the ranking of their scores.

Embedded methods [173, 174, 194, 195] aim to address the limitations of wrapper and filter methods by generating models while simultaneously selecting the features that will best contribute to their performance. The most common of these approaches are regularisation methods such as, Least Absolute Shrinkage and Selection Operator (LASSO) [196], Ridge Regression [197] and Elastic net regularization (which combines the first two) [198] along with many more. The computational complexity of the embedded method tends to fall between the wrapper approaches and the filter approaches.

Lashkari *et al.* [199] applied two feature selection techniques in their study. They used the WEKA¹⁵ DM software to select the key features from 23 time-based features to predict the service type of Tor traffic. The **first** technique (CfsSubsetEval+BestFirst) used the BestFirst search algorithm to select the best subset of features based on their evaluation criterion which employed the Correlation-based Feature Subset Selection (CfsSubsetEval)¹⁶ [200]. In the **second** technique (Infogain+Ranker), features were evaluated using the information gain before being ranked by their weights. The point at which the largest weight decrease occurred between two consecutive features was used as the cutoff point to select features with higher weights. This study showed that reduced feature sets were as able to build models with predictive power

¹⁵ <https://www.cs.waikato.ac.nz/~ml/weka/>

¹⁶ <http://weka.sourceforge.net/doc.dev/>

as full feature sets. However, in this study, feature selection did not take account of traffic variability as the data were divided randomly into training (80%) and evaluation (20%) sets.

In another study, Aljawarneh *et al.* [201] proposed a hybrid model composed of multiple base learners, where a vote algorithm with Information Gain was used to combine the probability distribution in order to select the salient features that increased the accuracy of the model as a whole. The selected features were then used to build multiple classifiers, where the best classifier (based, on its performance on the validation data) was selected. However, careful analysis of their work has revealed that a fixed threshold was used to select features with a weight higher than 0.4, which raises doubts over this approach under variable traffic patterns.

Ambusaidi *et al.* [202] proposed an algorithm to analytically select optimal features based on mutual information. The proposed algorithm has the capability to handle features with linear and nonlinear dependencies. Features selected by this algorithm have been tested on a Least Square Support Vector Machine based IDS (LSSVM-IDS) using different datasets (KDD Cup 99, NSL-KDD and Kyoto 2006+ datasets). The resulting feature subsets led to an increase in accuracy and a reduction in computational cost. However, the comparisons they made with other approaches was limited to the results and figures that had already been published of the compared state-of-the-art models and methods. Also, there was no statistical comparison to determine the significance of any differences.

In contrast to the studies which discuss the importance of feature reduction, and based on the observation of the high detection rates of the Maximum Entropy [203] and PHAD [45] detectors of portscan attacks, Ashfaq *et al.* [204] suggested using a high dimensional feature space to improve detection. They argued that limiting detectors to specific features could reduce their accuracy as a result of changing traffic patterns. However, this suggestion is limited to a specific family of attacks and generalising it requires careful testing, as extending the feature space could slow the detectors.

Ambroise and McLachlan [205] have pointed out the importance of undertaking the feature selection stage during the training phase when the Cross-Validation technique (discussed in **Section 2.3.1**) is applied. This is to avoid any bias in the selected features which would be mirrored in the accuracy of the predictions.

2.2.3.2 Data (class) balancing

The class imbalance problem corresponds to the phenomenon whereby classes in a training dataset are not equally represented [8, 206, 207]. This phenomenon is common in the domain of network ID, where normal traffic forms the larger majority class. Veeramachaneni *et al.* [98] estimated in an enterprise setup, which was based on the characteristics of the data ingested by their proposed platform, that attack cases form less than 0.1%. This irregularity can affect the learning capability of many ML and DM algorithms, resulting in predictive models that favour the majority class.

A number of solutions are proposed in the literature to address this issue, however, the most common ones are:

- **Data sampling methods**

- **Under-sampling techniques** reduce the number of samples in the majority class until they match the size of the other class(es) [206]. The most common under-sampling techniques are the Random Under-Sampling (RUS) [206] and Tomek Links (TL) [208]. However, many alternative under-sampling methods exist which employ various techniques, such as Clustering-Based [209, 210], Evolutionary Algorithms (EAs) [211] and Genetic Algorithm (GA) [212]. However, the under-sampling approach can lead to information loss, as part of the dataset will be wasted and not used.

- **Over-sampling techniques** work the opposite way; they increase the number of underrepresented (minority) class(es) to match the size of the majority class [206]. Randomly resampling minority instances and the Synthetic Minority Over-sampling Technique (SMOTE) [207] are the most common of these techniques. Since the introduction of the SMOTE algorithm, multiple variations of it have been proposed, such as Borderline-SMOTE [213] and safe-level SMOTE [214]. Other techniques use different procedures in their over-sampling methods, such as Mahalanobis distance-based [215] and density-based clusters (DBSCAN) [216]. He *et al.* [217] proposed an adaptive variation of SMOTE known as ADASYN, which generates more synthetic samples of minority instances which are more difficult to learn than the easier ones. However, over-sampling could slow the model learning task as the number of samples increase. Also, the SMOTE technique and its derivations could introduce noise into the data by generating non-representative samples, which may not contribute to the learning task.

- **Learning parameter control methods**

- **Cost function based approaches** are used to assign a larger cost (weight) to misclassified minority instances than those of the majority class. However, these methods have some limitations in addressing the imbalance problem in multi-class cases, due to the exponential growth of the parameter tuning space which increases with the number of classes [8]. Also the selection of the right cost value is subject to an iterative process and in many cases requires subjective decisions from domain experts.
- **One-class anomaly learning methods** are used to build a prediction model using one of the classes in the training phase [218-220]. These models will flag

any deviation from the learned class as an anomaly. Das *et al.* [8] highlighted the suitability of these methods for domains with very high imbalance ratios. However, as pointed out by Yu [221], to learn an accurate boundary for the required concept (class), these methods require a greater number of that concept's samples [222]. Also, the adaptability of these methods to changes in data over time requires further investigation.

Many other techniques were listed in the seminal review by Das *et al.* [8], such as *Boundary Shifting methods*, *Active Learning* and *Kernel Perturbation techniques*. The review also listed some hybrid approaches that combine different techniques, such as data sampling methods and controlling the parameters of the learning algorithms. Furthermore, the over-sampling and under-sampling techniques listed above, along with many others, can be found in various known software, such as a python package¹⁷ provided by Lemaître *et al.* [223] that encompasses all of these techniques.

An excellent systematic study was conducted by Japkowicz and Stephen [224] to compare different methods used to address class imbalance (over-sampling, under-sampling and cost-modifying) on different ML algorithms (C5.0, Neural Networks and Support Vector Machines). They concluded that there are four main factors that affect class imbalance: *the degree of class imbalance; the complexity of the concept represented by the data; the overall size of the training set; and the classifier involved*. In the study of Japkowicz and Stephen noted that, sensitive classifiers showed an inverse relationship between the size of the training set and the level of class imbalance, whereas the degree of class imbalance behaved positively relative to the complexity of the concept. The study showed that of the three algorithms, the C5.0 algorithm was the most sensitive to class imbalance, while SVM was the least sensitive. Finally,

¹⁷ <https://github.com/scikit-learn-contrib/imbalanced-learn>

they showed that over-sampling, which was more useful than under-sampling, improved sensitive classifiers, whereas it had the opposite effect on insensitive classifiers.

2.3 Evaluation of Intrusion Detection Systems

Various evaluation methods and measures have been proposed and used to assess the performance of prediction models. The following subsections present the main methods and measures that are commonly used to evaluate different IDS.

2.3.1 Evaluation methods

Evaluation methods describe the approaches used to carry out model learning and the testing tasks used to assess the performances of these models. Most of these approaches are related to the method used to split the data when such evaluations are undertaken.

Fielding and Bell [225] discussed the different data partitioning strategies used in experiments and evaluations. These strategies are:

- The **Resubstitution** method which uses the same dataset for both training and testing phases. According to the literature available, this method is not used (and should be prohibited) in many fields, such as Computer Science, because of its tendency to overfit and to report overly optimistic results.
- The **Prospective sampling** method is used by obtaining a new sample data after the model generation phase is over. This method is not a commonly used evaluation practice in anomaly-based detection.
- The **Bootstrapping** method is used to sample instances *with* replacement to create bootstrap samples. Each one of these samples is then used to build a predictive model, which will be tested on the remaining instances that were not selected. As recommended by Verbyla and Litvaitis [226], this process should be repeated many times, i.e. 200 or 1000 times, and the mean of all performances should be reported. Although this technique is a built-in function

of some algorithms, such as Random Forest and C5.0, it is not a very common method of assessing the performance of ML and DM models and techniques. This could be attributed to the large data sizes used in the ID domain, where model development and assessment incurs high computational costs.

- The **Randomization** method is used to sample instances *without* replacement to obtain random samples, so that the evaluation process and subsequent reporting follows the same approach as the bootstrapping method. Similar to bootstrapping, this method is not commonly used, as a variation of it is performed by undertaking multiple runs of the k-folds method, as discussed next.
- The **K-fold partitioning** method is the most common evaluation exercise in the IDS domain. It is performed by partitioning the data into K (usually $K > 2$) splits, so that K-1 sets can be used to build a prediction model and only one part is used to test the model [227]. This process is repeated for every partition in what is known as the K-folds Cross-Validation process so that the mean of all the K performances is reported. This method has two special cases:
 - The **Leave-One-Out (L-O-O) method** (also known as jackknife sampling) sets K to the number of samples (N). In this method, a model is trained on N-1 samples and tested on only one instance. It is common in this method to repeat the model generation and testing N times, which makes the process very slow when the number of samples is very large. As a result this method is not commonly used in anomaly-based IDS research.
 - The **Hold-out (external) method** sets $K=2$ so that the dataset will be split into two parts, with one part used for training and the other for testing. However, this partitioning does not have to be a fifty-fifty split as various strategies can be applied.

As the K-folds Cross-Validation process is commonly used to split data into K partitions at random, this process could result in partitions that are statistically similar to each other. Hence, possible variability in the patterns of network traffic over time will become distorted. This could explain the high performance of the different learning algorithms discussed in the literature. Further investigation into the effects of this method on concept drift is required and a comparison to other sampling methods, such as Prospective Sampling [225, 228], is needed.

In general, evaluating the effectiveness of an IDS depends on the availability of information about new or known intrusions. The difficulty in this is that it relies on human expertise to assess any potential security vulnerability and to provide the best response to such intrusions. As Corona *et al.* [26] highlighted, there are no standard methods to govern these types of evaluations.

Furthermore, these techniques are usually applied in a batch learning process, as real time IDS handle data as they flow.

2.3.2 Evaluation measures

This section provides an overview of the main evaluation measures and metrics widely used to assess classification models and algorithms.

2.3.2.1 Confusion matrix

Many performance assessments of classification models in a supervised learning task are computed using basic counts of a table known as a **confusion (error) matrix** [229]. Without restriction to the order, the columns in this matrix contain the number of instances of every actual class (label) in the classified data, while the rows contain the number of instances of every class as predicted by the model. **Table 2.1 (a)** and **(b)** show the structures of these matrices for binary and multi classification problems respectively, where the binary matrix is a special case of the multi-class table. **Table 2.1-(a)** presents four basic measures for binary classification:

- **True Positive (TP):** represents the number of positive (class 1) instances that were correctly predicted by the model to be positive (class 1).
- **True Negative (TN):** represents the number of negative (class 0) instances that were correctly predicted by the model to be negative (class 0).
- **False Positive (FP):** represents the number of negative (class 0) instances that were misclassified by the model to be positive (class 1).
- **False Negative (FN):** represents the number of positive (class 1) instances that were misclassified by the model to be negative (class 0).

Prediction \ Actual	+ve	-ve
	+ve	TP
-ve	FN	TN

(a)

Prediction \ Actual	c_1	c_2	...	c_n
	c_1	$c_{1,1}$	$c_{1,2}$...
c_2	$c_{2,1}$	$c_{2,2}$...	$c_{2,n}$
\vdots	\vdots	\vdots	\ddots	\vdots
c_n	$c_{n,1}$	$c_{n,2}$...	$c_{n,n}$

(b)

Table 2.1: Confusion Matrices. (a) Confusion matrix for binary classification which is a special case of multiple classes. (b) Confusion matrix for multiple classes.

These measures are then used to compute many other complex measures, which can provide a better assessment of the performances of the prediction models, such as, *Accuracy*, *Geometric Mean of Accuracy*, *F₁ score* (known as the Harmonic mean of Precision and Sensitivity) and *Matthews Correlation Coefficient (MCC)*, etc [229].

To illustrate this concept, the following example (**Table 2.2**) presents three confusion matrices for dummy data that contain 100 instances, ten of which are negative (class 0) with the remaining 90 being positive (class 1). The term cutoff is explained later in **Section 2.3.3**.

Prediction \ Actual	+ve	-ve
	+ve	90
-ve	0	0

(a)

Prediction \ Actual	+ve	-ve
	+ve	47
-ve	43	4

(b)

Prediction \ Actual	+ve	-ve
	+ve	0
-ve	90	10

(c)

Table 2.2: Confusion matrices of dummy data with 100 instances (negative = 10 instances and positive = 90 instances) at different cutoff (threshold) values. (a) Cutoff (threshold) is 0. (b) Cutoff (threshold) is 0.5. (c) Cutoff (threshold) is 1.

The following subsection presents some of the measures that can be derived from these confusion matrices.

2.3.2.2 Accuracy

In the literature, **Accuracy** is the most popular measure to assess model performance and to compare different models [229]. It basically measures the proportion of instances that are correctly classified (TP and TN) to the total number of instances in the dataset. **Eq.(2.1)** is used to compute accuracy for binary models and **Eq.(2.2)** is the general formula for a multiclass model. The accuracies of the example data presented in **Table 2.2** of every matrix (a, b and c) are 90%, 51% and 10% respectively. This measure forms a very poor measurement when an imbalanced dataset is evaluated, as illustrated in the above example. Although the first table has an accuracy rate of 90% it failed to detect any of the negative cases as all instances were predicted to be positive. This example illustrates the limitation of this measure in assessing model performance if the data used are highly skewed. More appropriate measures are available and are discussed next.

$$acc = \frac{TP + TN}{TP + FN + FP + TN} \quad Eq.(2.1)$$

$$acc = \frac{\sum_{i=1}^n c_{i,i}}{\sum_{i=1}^n \sum_{j=1}^n c_{i,j}} \quad Eq.(2.2)$$

2.3.2.3 Other common measures

The following evaluation measures are some of the most commonly used to assess the performance of binary classification results. Unlike the accuracy measure, these measures cannot be extended to multi-class problems. They are derived from the values in the binary confusion matrix, as in **Table 2.1-(a)**.

Sensitivity represents the proportion of positive (class 1) instances that are correctly classified [229]. This measure is also known as the *True Positive Rate (TPR)*, the *detection rate* or *recall* (Eq.(2.3)).

$$TPR = \frac{TP}{TP + FN} \quad Eq.(2.3)$$

Specificity represents the proportion of negative (class 0) instances that are correctly classified [229]. It is also called the *True Negative Rate (TNR)* (Eq.(2.4)).

$$TNR = \frac{TN}{TN + FP} \quad Eq.(2.4)$$

Precision is a common measure in information retrieval and in the document classification domain. It represents the proportion of instances predicted as positive (class 1) that are actually positive [229]. This measure is also known as the *Positive Predictive Value (PPV)* (Eq.(2.5)).

$$Precision = \frac{TP}{TP + FP} \quad Eq.(2.5)$$

The **F-measure** is another common measure and is calculated based on the combination of precision and recall measures [229]. However, Hand and Christen [230] recently revealed a major conceptual weakness in this measure and urged researchers to find an alternative. This measure is also known as the *F₁ score* or the *Harmonic mean of Precision and Sensitivity* (Eq.(2.6)).

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall} = \frac{2TP}{2TP + FP + FN} \quad Eq.(2.6)$$

2.3.2.4 Geometric Mean of Accuracy

The **Geometric Mean (G-Mean) of Accuracy** [231] metric aims to address the limitations of the normal accuracy measure when dealing with imbalanced datasets. It measures the geometric

mean of the accuracy of each class. This measure computes the classification accuracies of every class separately and then computes their geometric mean. For binary classification, **Eq.(2.7)** is used to calculate the G-Mean accuracy. **Eq.(2.8)** shows the general formula used to compute this measure for multi-class problems, where $c_{a,b}$ is the number of class b instances that were predicted as a and n is the total number of classes. Calculating the G-Mean Accuracy of the examples in **Table 2.2** (a, b and c) produces the following results: 0%, 46% and 0% respectively. Both tables (a) and (c) that were classifying all instances into one class have attained a 0% G-Mean Accuracy as one of the other classes had zero accuracy. It is also worth noting that even table (b) scored less than the accuracy measure. If a model predicts all classes perfectly, then the G-Mean of Accuracy will be one.

$$gAcc = \sqrt{A_{+ve} \times A_{-ve}} = \sqrt{TPR \times TNR} = \sqrt{\frac{TP}{TP + FN} \times \frac{TN}{TN + FP}} \quad Eq.(2.7)$$

$$gAcc = \sqrt{\prod_{i=1}^n \frac{c_{i,i}}{\sum_{j=1}^n c_{j,i}}} \quad Eq.(2.8)$$

Although this measure was first proposed by Kubat and Matwin [231] few studies have used it to assess and compare the performance of different models. However, a number of recent studies in network ID domain have started to use it [232-234].

2.3.3 Threshold related measures

Almost all ML algorithms used for classification problems can return predictions in two forms; by predicated class (label) or by probability (score). When instructed to return the class label, the label of the class with the higher probability will be returned. Whereas if the class probability is returned, the user can vary the discriminating cutoff (threshold) at which class

labels are assigned. For example, in a binary classification problem {attack, normal} a model might return a probabilities vector for a tested instance as (attack:0.35, normal:0.65). If the cutoff value is set to default i.e $\text{attack} \geq 0.5$, then this instance will be classified as normal. However if this threshold is set lower, such as $\text{attack} \geq 0.32$, then it will be labelled as an attack. The cutoff (threshold) value is usually varied to maximise some measurement such as the prediction accuracy, or to minimise another, such as false rates. Varying the cutoff (threshold) value will generate different values in the confusion matrix as instances change classes accordingly, and the measurement result can be calculated at every cutoff point. This varying process is usually undertaken at the training phase to set models parameters, i.e. the cutoff (threshold) value, and to assess and compare model performances using different evaluation measures, such as the Receiver Operating Characteristic (ROC) and the Area Under the ROC Curve (AUC). Beguería [235] suggested the use of these two measures to address the dramatic effect of the class imbalance problem on many common validation statistics such as the confusion matrix and accuracy.

This is illustrated by the example presented in **Table 2.2** which shows the classification results of a dummy model with different cutoff (threshold) values: 0, 0.5 and 1. **Figure 2.1** presents these tables as graphs so that the prediction (on the y-axis) is the probability that an instance is positive (class 1), while the actual (on the x-axis) is the true label of that instance. Cutoff (threshold) values *zero* and *one* are the extreme cases at which instances will be classified into one class or the other.

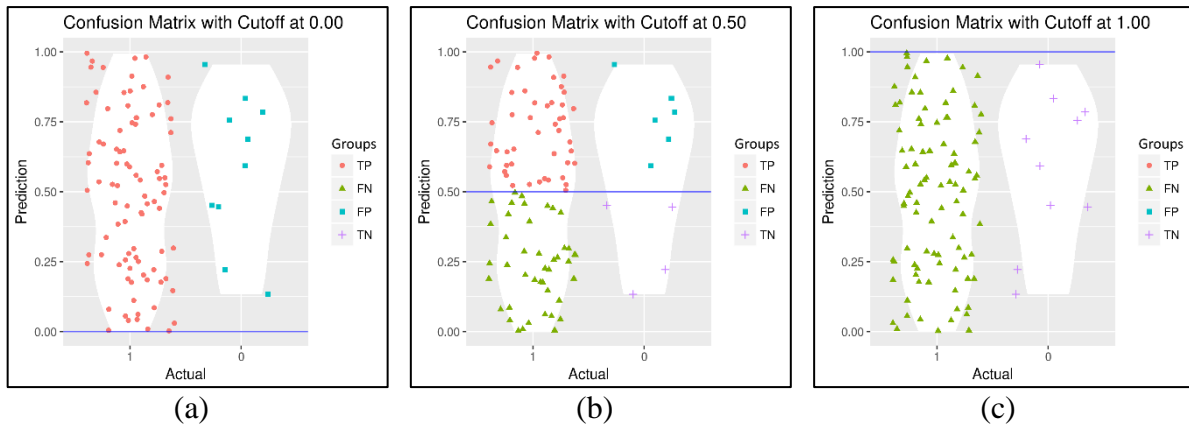


Figure 2.1: Plot of the confusion matrices for dummy data with 100 instances (negative (class 0) = 10 instances and positive (class 1) = 90 instances) at different cutoff (threshold) values. (a) Cutoff (threshold) is 0. (b) Cutoff (threshold) is 0.5. (c) Cutoff (threshold) is 1.

2.3.3.1 Receiver Operating Characteristic (ROC) Curve

The **Receiver Operating Characteristic (ROC) Curve** [236] is a graphical representation of a model's performance at different cutoffs (thresholds). It plots the False Positive Rate (FPR) (x-axis) against the True Positive Rate (TPR) (y-axis) for every discriminating cutoff (threshold) value used to assign instances to their class (see **Figure 2.2**). A model with good prediction ability will have a ROC curve that goes towards the upper left-hand corner, so that the closer this curve gets to the top left-hand corner, i.e. point (0.0, 1.0), the better its performance [237]. The closer the curve is to the diagonal dotted line, i.e. line (x=y), the worse the model will be, as it will be no better than a random guess.

Figure 2.2 provides an example of two different ROC curves for two different models. The ROC curve of a dataset with 100 instances (D_{100}) shows a very weak model as that curve is very close to the dotted line. These data points are the same as those presented in **Table 2.2** and **Figure 2.1**. The ROC curve for the data with 1,000 instance (D_{1000}) shows a much stronger performance as this curve is further from the dotted line.

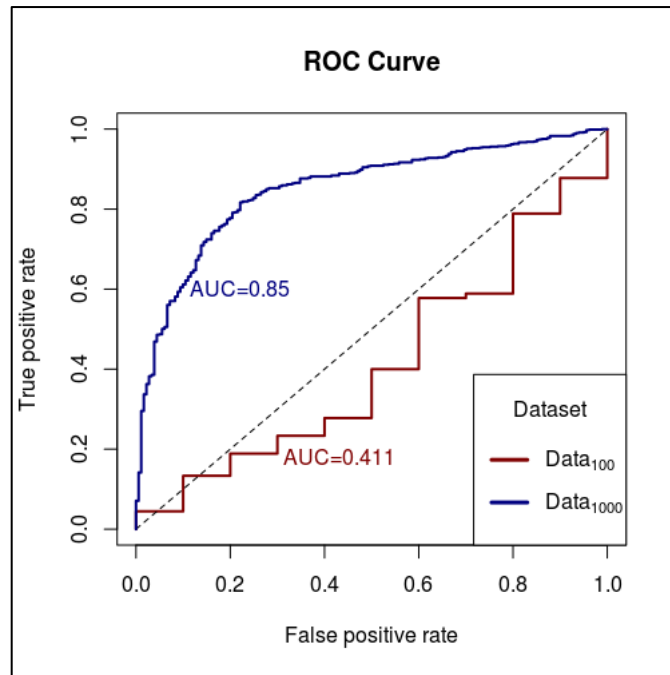


Figure 2.2: ROC curves for two dummy datasets with a different number of instances; D_{100} has 100 instances while D_{1000} has 1000 instances. For each dataset 10% of its instances are negative (class 0) while the remaining 90% are positive (class 1).

2.3.3.2 Area Under the Curve (AUC)

The **Area Under the Curve (AUC)** is used to measure a binary model's performance based on calculating the area under the ROC curve [229]. It provides a summary of the ROC curve and measures how well a model can distinguish between two groups [(positive, negative) or (attack, normal)] [238, 239]. However, this summary can lead to a loss of information about the trade-off between the True Positive Rate (TPR) and the False Positive Rate (FPR). In a perfect classification the AUC will reach one, whereas a poor classifier will have an AUC value of around 0.5, as the TPR increases linearly with the FPR [240]. Another key feature of this measure is its independence of the proportion of classes in the data, which makes it immune to the imbalanced data problem [240]. Therefore, AUC is commonly used to compare the performance of multiple models, as it can assess the discriminative power of each prediction model [241]. However, many researchers have criticised its use for model comparison as it is noisy when used as a classification measure [242-244]. Instead, they suggest interpreting AUC as a way to assess the probability that a model will rank a randomly chosen positive instance higher than a negative one. As illustrated by **Figure 2.2**, the dummy model developed using

dataset D_{100} has an AUC of 0.411, which is very close to 0.5, while the one trained on data D_{1000} attained an AUC of 0.85.

2.4 Related Work and Research Gaps

At an early stage, researchers realised that the performance of IDS were tightly related to the behavioural patterns of users as well as the characteristics of various underlying services and protocols. Therefore, anomaly-based methods were introduced to address possible deviations from normal behaviours in order to flag intrusions. In addition, many researchers understood that for various reasons these anomaly-based methods suffered from high false alarms. The key reason for this was their inability to adapt themselves to changes in data patterns over time. As a result, many proposals have been put forward to address this issue, including methods that adapt to such changes. Hence, many studies have suggested various approaches, such as model updating and rule tuning techniques. Many others have looked into the benefits of using adaptive or tuneable thresholds for the IDS measures to flag anomalies rather than relying on fixed thresholds. The following subsection presents the key work in this area.

2.4.1 *Threshold adaptation*

Lucchetti [245] stressed the importance of the continual tuning of system rules in the context of market practices, in order to be able to identify potential new risks. This scenario shares an important feature with network traffic, in that patterns are continually changing over time, which requires models to be adaptable [31]. Hence, Chen *et al.* [246] suggested undertaking threshold tuning for the predictions of classification methods that generate a quantitative output (score), so that the threshold can be set at different values to assign class labels. As a result, Catania and Garino [31] suggested performing tuning on statistical-based models whenever a change in network traffic patterns is detected by making adjustments to the normal model.

In an attempt to understand the importance of the right threshold selection on the performance of prediction models, Freeman and Moisen [247] investigated 11 optimisation criteria on

threshold selection by assessing the prevalence and kappa¹⁸ for the data of 13 tree species. They concluded that the species that were the most sensitive to threshold selection either have a low prevalence or a poor model quality. These findings can be projected onto the situation in network traffic, where, usually, anomalous traffic forms a minority compared to normal traffic and many anomaly detection methods have been developed based on this core assumption. Furthermore, due to the evolving nature of traffic, the quality of detection models tends to deteriorate over time. Therefore, threshold adaptation could help to improve the quality of these models before they get phased out.

To address model tuning, the conventional (batch-learning) modelling process usually has two main phases, training and testing. At the modelling stage, training (learning) data are used to build a prediction model, which is then used to predict the test (evaluation) data. However, Buczak and Guven [27] stressed the importance of having three phases, and introduced an intermediate (validation) stage. In this three phase setup, Buczak and Guven [27] suggested that the training data are used to build multiple models using different ML/DM algorithms with different parameters. The validation data could then be used to select the best model(s) and to estimate their errors before they are used to predict or classify the testing data. Buczak and Guven [27] recommended that the selected model should not be fine-tuned (model parameter tuning) based on how it performed on the test data, to avoid reporting overly optimistic results i.e. reporting accuracy rates that might not be true for another test dataset. Although the recommendation not to modify model parameters based on the test data is a reasonable design practice, many of the recommended adaptive real-time systems (see **Section 2.4.1.2**) perform tuning on detection rules. Therefore, threshold tuning based on the prediction scores of a model could provide a tool to tune the system over time. However, the single fine-tuning recommendation may not be appropriate as it appears to be based on the assumption that test

¹⁸ The Kappa measure is used to assess the inter-rater reliability (agreement) between two raters (classifiers) [386].

datasets, including future unseen data, have similar statistical properties; which is not a valid assumption given the variable nature of network traffic.

Similarly, in an attempt to find the right discriminating threshold for the detection model, Beguería [235] suggested the use of validation data. The selected threshold is then used to classify the records in the evaluation/test data based on their scores returned by the prediction model. However, this suggestion does not appear to take into account the variability of behaviour in input (traffic) data over time, much like the single fine-tuning recommendation by Buczak and Guven [27].

Most of the model tuning and adaptability to pattern changes in network traffic in the field of ID can be categorised into three main themes which are outlined below.

2.4.1.1 Batch learning

Yang [248] proposed score-based local optimisation (SCut) as a strategy to select a threshold based on optimising a performance measure, such as accuracy. In other words, SCut is the threshold at which a performance measure would be maximised or minimised. To the best of the obtained knowledge, no studies have explored model adaptation for changes in network traffic by tuning the threshold of the predictions of a model within a batch-learning setup.

Lakhina *et al.* [249] used the Principal Component Analysis to separate a high-dimensional space of network traffic measurements into disjoint subspaces. Each subspace corresponded to normal or anomalous network settings. They used a fixed threshold (3σ deviation from the mean) to separate the principal axes into normal and anomalous sets and found that the first four principal components represented the normal subspace for the cases they analysed. This study did not address the variability of traffic over time, and so requires further analysis of the performance when traffic conditions vary.

In an attempt to investigate the effect of threshold tuning on multi-class predictions, Fan and Lin [250] concluded the effectiveness of tuning approaches on the performance of classification techniques. They used the 5-folds Cross-Validation technique to evaluate these effects, despite the fact that, as discussed earlier, the Cross-Validation technique may not maintain any statistical differences between the training and the testing data (**Section 2.3.1**), leading to overly optimistic results. They analysed the effect of different optimisation metrics (macro-average F-measure, micro-average F-measure and exact match ratio) on the overall performance of the selected threshold. They then investigated this tuning approach using validation data without considering whether such tuning was required for every independent evaluation process or whether the selected threshold could be used for future evaluations performed by the prediction model. Similarly, Pillai *et al.* [251] investigated the issue of threshold selection for multi-label classification problems by optimising the F-measure and Precision-Recall curve. They used 5-folds Cross-Validation on five datasets to validate their results. They compared the results obtained on the evaluation/testing data by using the optimal threshold that had been selected on the basis of the validation data. However, they did not extend their analysis to compare their results with those where the threshold had been tuned for the testing data. They concluded that selecting an optimal threshold based on maximising the micro-F measure can lead to overfitting.

Koyejo *et al.* [252] investigated the optimisation of a binary classifier using different metrics where they proposed an approach to identify the optimal threshold based on the conditional probability of the positive class. However, in this approach the search for the optimal threshold was performed using training and validation data. Yan *et al.* [253] pointed out that this search requires prior knowledge of the optimal classifier, which is usually unknown in reality. As a result, Yan *et al.* [253] identified two key properties (Karmic property and the Threshold-Quasi-Concavity property), for which they have shown, theoretically, that the Bayes optimal classifier is a threshold function of the conditional probability of a positive class. On

this basis, they proposed a novel threshold estimator. As with previous approaches, these works do not seem to assume a change in data over time (concept drift), as the threshold is only set once using the validation set. In general, nearly all approaches in the batch-learning methods, adopt the recommendations of using a single validation dataset to select the right threshold as suggested by Beguería [235] and, Buczak and Guven [27], where this adaptation is only applied once.

2.4.1.2 Real-time learning

In an early study, Eskin *et al.* [254] proposed an adaptive Host-based ID model generation. Their framework, which is similar to that of Honig *et al.* [255], recommends the aggregation of all data, such as system calls collected by sensors from every monitored host, into a single data warehouse. This data can then be used to train detection models, which can in turn be distributed to hosts to detect intrusions. The adaptability of this framework is in the deployment of models on the hosts. However, this framework uses a fixed threshold to flag anomalies without addressing the variability between the hosts. Also it shows a scalability limitation, as with the amounts of data generated by the monitored hosts, storing such data will become a serious issue over time.

Hossain and Bridges [256] proposed a framework for adaptive IDS using fuzzy Data Mining. This framework aims to minimise the human intervention in the adjustments of the profiles used to describe normal traffic by the IDS. The tuning process is designed to operate on a real-time IDS. Hossain *et al.* [257] evaluated this framework by using a sliding window to update the profile, so that the updating process used the data that fell within that time window. Some heuristics were used to decide when the updating process should be triggered. For their analysis they used a 10 week capture of real network traffic from the Computer Science Department at Mississippi State University. Within this period some simulated *portscan* attacks were performed. There are no details in the study about the nature of the collected traffic. Also, it

seems that they considered all traffic, other than those *postscans*, as benign. These experiments were designed to detect known attacks, however the system produced results that the authors could not explain, which could be attributed to the lack of controls over the traffic that was analysed.

Jung *et al.* [258] developed a Threshold Random Walk (TRW) algorithm to detect random *portscan* attacks in a real-time setup, based on the observations of the state (successful or unsuccessful) of connection attempts from a remote host to newly-visited local addresses. They modelled accesses to the monitored systems using a random walk on one of two possible sets of probabilities, which were specific to their detection principle. Each of these two probabilities was used for one of the boundary thresholds (i.e. the lower and upper thresholds). However, this model assumed that all distinct connection attempts had the same likelihood of success, while no correlation between these attempts was assumed. Further, as Ali *et al.* [259] pointed out, threshold adaptation was only performed on the upper boundary of the likelihood ratio which was based on previously observed instances, while the lower boundary was fixed.

Idé and Kashima [260] investigated the development of an IDS to detect anomalies in multi-tier systems, such as web-based systems. They modelled the system using a weighted graph, so that the service activities were subsequently used to extract a feature vector, which was computed using the principal eigenvector of the eigenclusters of the graph. They defined an anomaly measure by using a derived probability distribution i.e. an approximation of the von Mises-Fisher distribution, where at a critical probability boundary the threshold value was adaptively updated. As this IDS models service activities in the system, where the directions of these activities are assumed to be stable, services which are rarely used may not benefit from its detection capabilities. As a result, services run by careful adversaries, such as Command-and-Control (C&C) might not be flagged up.

Yu *et al.* [261] proposed an automatically tuning IDS (ATIDS) system, which used feedback from the security officer about encountered false predictions to automatically tune the threshold of the rule-sets of their rule-based prediction model in real-time. This system is dependent on the human resource available, which might impose a challenge when security officers are overwhelmed by alarms. Therefore, Yu *et al.* [262] proposed an extension that adjusts the number of alarms flagged to security operators based on their abilities. Although this extension minimised the burden on security officers, the overall performance of the system was limited by the time it took to provide feedback. That is, delayed feedbacks could hamper the performance due to the delayed adaptation. This system also failed to cope with drastic changes in system behaviour, as the tuning process is performed on the rules level of the detection model, where these rules set might not be representative of the new behaviour due to concept or feature drift (see **Section 2.1.4**).

An outstanding study by Ali *et al.* [259] proposed a generic threshold tuning algorithm so that the detection threshold of any score-based Anomaly Detection Systems (ADS) could be adapted. In their approach, statistical and information theoretic analyses were undertaken on the anomaly scores produced by multiple network-based ADSs (*PHAD* [45], *Maximum-Entropy* [203], *Sequential Hypothesis Testing* and *PAYL* [46]) and host-based ADSs (*Anomalous Sequence Detector* [263], *a Machine Learning based Detector* [264] and *Sequence Alignment based Detectors* [265, 266]). These analyses aimed at revealing consistent structures of time correlation during periods of normal activities. They used Markov chains to model the observed time correlation structure, and these Markov chains used a stochastic monitoring framework to tune thresholds for the detection of the ADS as per the real-time measurements. In an attempt to protect the system from sporadic changes and evasion attacks, and to enhance its resilience, some statistical techniques were used. However, this approach targeted anomalies that cause a detectable variability in traffic patterns due to their high volume, such as

UDPFlood, *TCP SYN Flood* and *TCP SYN portscans* attacks. This approach is designed for score-based **real-time detectors** (not batch) as they quantify the anomaly score based on a comparison between the learned profile and the run-time profile.

In an attempt to keep up to date with the latest technological advancements, Chou and Wang [267] proposed an adaptive network IDS for Cloud environments. They claimed that their system had the capability to perform automatic labelling of raw network traffic (normal and anomalous). This claim was based on the fact that they had used a spectral clustering algorithm (unsupervised learning) to cluster the unlabelled network traffic so that the clusters could later be used as labels to construct a decision tree-based detection model. As the spectral algorithm clustered the incoming traffic, these clusters (labelled data) were used to improve the original detector and to adapt it to the network environment by building a new detector. Although the authors claimed that this system was developed for Cloud environments, they used DARPA 2000 and KDD 1999 datasets in their experiments, without any justification as to why such data had been selected for this scenario. They also proposed an unsound experimental design which overlooked any DDoS attacks in DARPA 2000, claiming that this type of attack would generate lots of connections, fearing that it could defy the core assumption of the rarity of attacks. This decision calls into question how their system would perform in a real life setup as such assumptions are not guaranteed in production environments. It also appears that they based their study on the assumption that the patterns of the attack traffic were different and as such, could be separated out from those of normal flows, as the decision tree model was developed on the clusters formed by the spectral clustering algorithm.

Agosta *et al.* [268] introduced a distributed Anomaly Detection System (ADS) to detect worm threats. This system employed a threshold adaptation technique, to compare it with the performance of a fixed threshold. This study concluded that the adaptive threshold technique

was far superior. However, these techniques were specifically designed for this type of attack and the ability to generalise these results to other class of threats is debatable.

Gu *et al.* [269] devised a framework to measure the effectiveness of IDS quantitatively. This method is based on quantifying the *feature representation capability*, *classification information loss* and *overall Intrusion Detection capability* of an IDS using a set of information-theoretic metrics. These metrics perform fine-grained evaluations of IDS and offer an assessment tool to compare multiple IDS to their specific components, not just based on their overall performances. Gu *et al.* discuss the importance of dynamic fine-tuning over static fine-tuning to address the issue of traffic variability over time. Thus, their framework introduced dynamic fine-tuning by dividing the time series into a number of intervals so that the tuning process is performed based on maximising the ID capability for each interval. However, Strasburg *et al.* [270] have raised concerns about the practical effectiveness of such a model in IDS development.

Jyothsna and Rama Prasad [271] studied a meta-heuristic assessment model, which aimed to set a threshold for random normal behaviour in real-time, by estimating the degree of intrusion scope threshold from a given network transaction. At the same time the model aimed to identify any new intrusions in the network. Feature selection based on feature correlation methods were performed to reduce processing and time costs. However, this approach did not cater for the effect of concept drift on the selected features over time, and hence, on a model's performance.

2.4.1.3 Data stream learning

In the counterpart techniques (data stream) to batch modelling methods, concept drift is a core feature that is considered in the modelling process. Therefore, in their seminal work Bifet *et al.* [272] proposed a new data stream framework which aimed to address concept drift by employing ensemble methods using various Bagging techniques. They later developed this framework into an open source software known as *Massive Online Analysis* (MOA) [143].

Masud *et al.* [273] proposed a classification method to address concept drift in data classes, that is, the emergence of unseen classes (labels). This is because, usually, new class labels require a longer time to be provided with new training data to rebuild the base detection models. Therefore, Masud *et al.* applied some clustering concepts to measure the distance between known classes and new data instances so that this technique could flag up these new instances as anomalies. However, Farid *et al.* [274] stated that such models would need to gather a large number of test instances to determine their similarities and differences in order to identify any novel classes.

In an earlier study in the same line of research, Masud *et al.* [275] proposed another detection approach for novel classes that used an adaptive threshold and the Gini Coefficient for outlier detection. For every classification model, the adaptive threshold technique defined a slack space outside its decision margin. Hence test instances that lay beyond that slack space were considered outliers, otherwise they are considered of the same class. These outliers were further tested using the discrete Gini Coefficient to determine whether they were noise or a novel class. However, the proposed approach is unable to distinguish between the novel classes if multiple new classes have emerged, as well as it does not cater for other types of evolution, such as feature drift [276].

In order to automatically determine the optimal parameters of an anomaly detector (AD) Cretu-Ciocarlie *et al.* [277] enhanced the training phase by introducing a self-calibration stage. Their method consisted of applying ensemble methods to unsupervised learning techniques to build micro-models. A weighted voting scheme on labels returned by these micro-models was used to compute a final class decision. In this method, automatic adaptation of the voting threshold is performed, where this threshold measures the degree of strictness or relaxation of the system by defining the minimum number of votes needed to accept the packet being tested. However, this approach could result into an Anomaly Detector (AD) that might be subject to attack as an

adversary could train it. This approach seems to fail to differentiate between a real change in traffic patterns and an ongoing crafted attack aimed at skewing the majority votes of the micro-models.

Chen *et al.* [278] suggested the offline mining of an old data stream to build high-quality models for every recurrent concept. When concept drift is later detected in a data stream, it could then be evaluated to identify the type of concept so that the traffic could be passed to the most suitable pre-built model to classify the traffic in that stream. This technique claims to achieve high rates of accuracy because of the high-order models. However, it assumes that there is a finite number of concepts to be modelled. This assumption is challenged by the high volume and diversity of network traffic. In addition, as the number of concepts grow over time this could form a bottleneck to the scalability of the system.

In a more recent work, Gomes *et al.* [279] proposed an Adaptive Random Forest (ARF) algorithm that was suitable for evolving data streams. This algorithm has the potential to address concept drift by adapting itself to any changes. The adaptation is performed by replacing any outdated trees in the forest with new trees that have been grown (trained) in the background.

2.4.2 Research gaps

As presented, the importance of adaptation to pattern variability has mainly been addressed in the context of real-time and data stream problems. Most of the adaptation and tuning approaches for **real-time** based systems target certain classes of attack which are formed of abrupt patterns, such as DoS attacks. As these attacks introduce high variability into traffic patterns, much research has attempted to detect them and fine tune the system accordingly. In most cases, these tuning approaches would aim for adapting the IDS detection parameters to increase or decrease thresholds of these parameters. However, Catania and Garino [31] pointed out that most of the adaptation approaches are aware of the high network variability and the proposed methods

provide the required adaptability features to adjust for the targeted anomalies. Similarly, in the **data stream** field, most of the proposed approaches suggest building new detection models to adapt to such changes [259].

As for the **batch-learning** tasks, in an ideally designed experiment, adaptation is undertaken only once for the prediction model using validation data [27, 280]. Validation data is used to estimate class distributions in order to calculate the optimal threshold for the prediction model. However, in a real life setup these distributions are not fixed, which renders such approaches ineffective. Furthermore, using a fixed threshold for predictive models could result in an inaccurate reading of the model's performance which could in turn lead to the selection of weaker models or an early phasing out of good models. However, no study exists to investigate continuous adaptation for every evaluation/test data based on the ground truth of a representative sampled subset to be used as a validation data to set the threshold accordingly. Therefore, further investigation is needed to study such an approach and to examine the effect of the size of sampled validation data on the overall performance of such an adaptation.

Moreover, in **batch-learning** approaches, there is a reliance on the K-folds Cross-Validation technique to evaluate models and when attempting to address the pattern change problem, validation data is the alternative suggested approach. Such an approach is used to select the best threshold based on the optimisation of some measure, such as the accuracy, for the prediction model. However, no study has investigated how a fixed threshold will behave under different setups. Also, as model development is based on various decisions taken in relation to the training data (such as feature selection and data balancing), it is important to analyse how such decisions might affect the model performance when traffic changes over time and causes concept or feature drift. It is also important to address whether the threshold (tuning) adaptation of model predictions have any effect on eliminating or mitigating such limitations.

This chapter also listed commonly used datasets to evaluate various IDS; most of the recent studies still rely on the KDD cup 1999 and NSL-KDD datasets, due to the lack of suitable alternatives. Although, more up to date datasets are available, most of them are in a raw format making them unsuitable for ML and DM learning tasks. As such, there is a need to transform an up to date dataset following a clearly defined, validated and reproducible transformation process.


2.5 Summary

This chapter has presented an overview of the different types of ID and the different methods used in the ID domain. It has also listed the datasets most widely used to evaluate ID as well as the various evaluation methods and the most commonly used measures. This chapter has highlighted the importance of model adaptation/tuning to address the evolving nature of network traffic in order to maintain acceptable levels of detection performances. It has also shed light on the latest developments in this area of research and identified a number of gaps that need to be addressed and investigated further.

The aim of this thesis is to investigate the effect of discriminating threshold adaptation on the accuracy (i.e. the Geometric Mean of Accuracy) of score-based anomaly ID models in batch-learning setups. This adaptation will be analysed and evaluated under different scenarios using three different types of ML algorithms. As this threshold adaptation is expected to improve the accuracy of the ID models, it should provide an accurate reading of the optimal accuracy of the detections of these models. This will result in maintaining the ID models for a longer time as they will be phased out less frequently, and hence save valuable resources.

Chapter Three

Experimental Overview



After previously discussing the current field of network ID and exploring the main problems in this area around open research, some of the key research gaps are now addressed in this thesis. This chapter introduces the research strategy and the empirical techniques employed in the experiments conducted to address some gaps.

The hypothesis underlying this research is, “*in a binary batch-learning setup, prediction accuracy of a score-based anomaly intrusion detection model can be improved by adapting the discriminating threshold specifically for the predictions of the evaluated network traffic.*”.

In order to address this hypothesis, three main experiments were conducted. Each experiment aimed to provide a deeper insight into the research problem by providing an empirical analysis for one aspect of the main research question. Three different ML algorithms (*C5.0*, *Random Forest* and *Support Vector Machine*) were evaluated and analysed using various datasets and model development setups.

Although each relevant chapter provides a detailed description of the data and methods used in the experiments, this chapter provides an overview of those methods in addition to discussing some of the experiments’ common features.

The chapter is divided into four main sections: an overview of every experiment; an overview of each of the ML algorithms used in the experiments; a brief description of the experimental methods and analyses used; and finally, an outline of the main limitations, alongside a summary of the chapter.

3.1 Overview of Experiments

As discussed in **Chapter 2**, there are various ways to adapt prediction models to variability in network traffic, such as in real-time and data stream methods. However, batch-learning setups are the least researched in that domain, although they are important to detect novel attacks that cannot usually be detected by other methods. Some kinds of attacks are better detected in a batch mode to increase the detection rate rather than attempt faster detection in real-time with a higher failure rate. With this approach, there is no need to change or tune any of a model's parameters as long as its predictions are in the form of a probability score. In this sense, threshold adaptation does not require any modification to the anomaly detection model. The detection model is thus treated as a black-box, as the adaptation is performed to its predictions and not to its detection parameters.

This section provides an overview of each of the main experiments which are outlined in more detail in the coming chapters. **Figure 3.1** illustrates the sequence, and the relationship, between the various chapters that detail the experiments.

3.1.1 *Experiment 1*

The first experiment examined the effect of threshold adaptation on the overall performance of a detection model. It set out to provide a Proof of Concept (PoC) by comparing three well-known ML algorithms to determine which was the most adaptable, that is tuneable. This experiment used the same datasets and the same ML algorithms in two different kinds of setup, each of which reported different results.

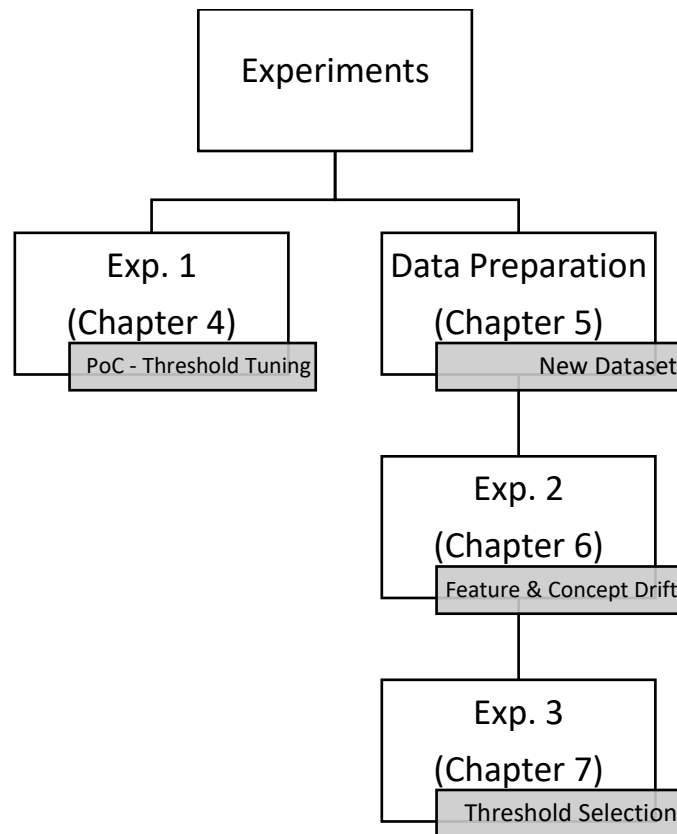


Figure 3.1: Schematic overview of the experiments undertaken

The first setup employed a 10-folds Cross-Validation to train and test the prediction models. The second setup used a prospective sampling approach to build models by using a 10-folds Cross-Validation on a subset of the data so that the models generated could be used to evaluate data with similar and, separately, differing statistical properties (concepts). The performances of the models in the second setup were compared before and after adapting the discriminating threshold for the evaluation data.

3.1.2 Experiment 2

The usual model development could be governed by some decisions made to improve some performance measures, i.e. speed or detection rate. Such decisions, which might involve executing a feature selection and/or a data balancing stage, are usually based on the analysis that will be conducted on the training data. As such, when new evaluation data are used the performance of the models may not be satisfactory, leading to a phasing out of those models

and the generation of new ones. However, such models may still be able to maintain high performances if they are adapted to the new concept that is introduced in the new data.

The discussion of this thesis considers threshold adaptation and investigated which of the three ML algorithms was the most adaptable to changes in the network patterns given that pre-decisions (feature selection and data balance) are taken based on the training data. Those decisions provide different setups for the new data rather than those used to build the model.

3.1.3 Experiment 3

The aim of this experiment, in this thesis, is to demonstrate the importance of adapting the prediction threshold for every individual evaluation dataset to maximise a model's performance. It investigates whether such threshold adaptation can be performed based on validation data that is sampled from the original population of the evaluation data. This third experiment therefore investigated the effect of the size of the data sample and the sampling technique in determining the optimal threshold for the evaluation (test) data. As such, the validation data will require the knowledge of the true labels of its samples; this experiment therefore investigated the effect of introducing different error levels into the true labels to assess the effectiveness of threshold tuning under such conditions.

3.2 Overview of Classification/Machine Learning algorithms

For all of the experiments conducted in this thesis, three common classification algorithms that are widely used for batch-learning were analysed, evaluated and compared, to address the anomaly network detection. These algorithms were C5.0, Random Forest and Support Vector Machine (SVM); this section provides an overview of each of these algorithms.

3.2.1 Decision Trees (C5.0)

C5.0 is a classification algorithm [281] based on decision trees. It is an improved version of the well-known C4.5 algorithm and addresses many of the latter's limitations. In comparison to its

predecessor, the C5.0 algorithm has a lower error rate due to its use of ‘boosting’, a technique to aggregate the results of multiple weak models (trees) to form a strong one [282]. Also, as C5.0 generates smaller trees, it consumes less resources, such as memory, and performs faster execution. Unlike C4.5, the C5.0 algorithm also avoids overfitting noisy data [283].

Decision trees are used in classification problems to build a deterministic data structure that is formed of decision rules for a particular domain [284]. Such trees are composed of nodes and leaves. At the node level, a decision is made to split the data into two subsets based on a single feature (different splitting criteria are discussed next). Each subset is then used to build a subtree. In contrast, at the leaf level, the final classification decision is based on the path traversed from root to leaf; these decisions can be either, a ‘class’ (label), or ‘probabilities’ (score) of classes as illustrated in **Figure 3.2**.

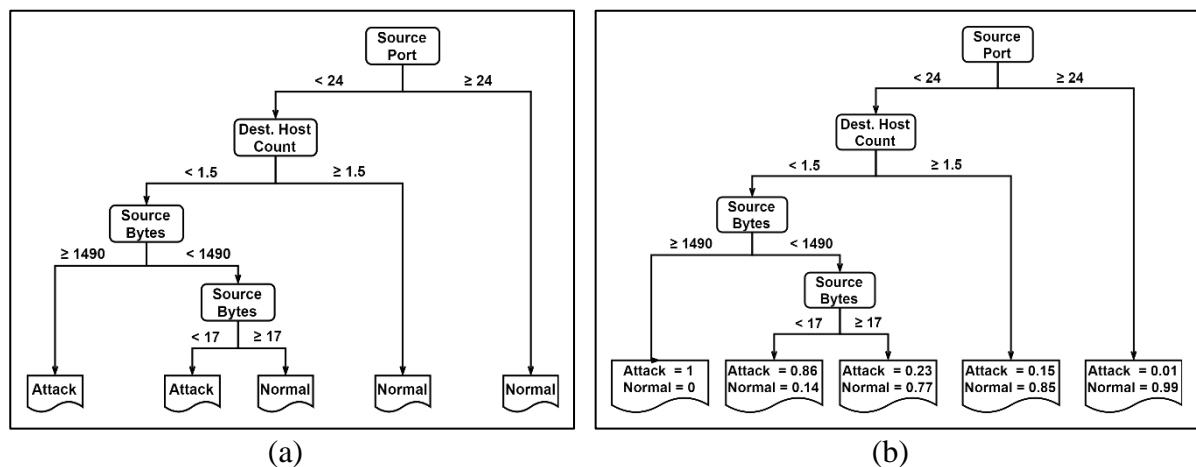


Figure 3.2: Example of a decision tree of dummy network traffic data with two classes {attack and normal}. (a) Returns the class label. (b) Returns the probability of classes.

3.2.1.1 Tree splitting criteria

The performance of different decision trees is tightly related to the splitting criteria used in building these trees. The main aim of these splitting processes is to maximise the purity of the classes in the subsets. In other words, **purity** measures how well the classes have been separated after the splitting rule has been applied. Many different metrics are used to measure purity, such

as the *Gini impurity* [285, 286] or *information gain* [287] for classification problems, and *mean square error* [285] or *variance reduction* [288] for continuous variables, *etc.*

Gini impurity measures the probability of misclassifying a randomly selected instance from a dataset if the distribution of labels (in the subset) has been used to randomly classify this instance [285]. This measure reaches zero when all instances within a subset are of the same class (pure). It is computed using **Eq.(3.1)**.

$$I_G(p) = 1 - \sum_{i=1}^J p_i^2 \quad \text{Eq.(3.1)}$$

where, p_i is the proportion of instances labelled as class i , and J is the total number of classes.

Information gain is used to select (recursively at each sub-tree) the best feature at which to apply splitting; the best feature being the one that provides the highest information gain [285]. This will ensure that relevant features are evaluated near the root of the tree (the top). This measure uses an entropy concept to compute purity, which is referred to as information. **Entropy** computes the degree of randomness of classes, or, in other words, it measures impurity, which is computed with **Eq.(3.2)**.

$$H(T) = I_E(p_1, p_2, \dots, p_J) = - \sum_{i=1}^J p_i \log_2 p_i \quad \text{Eq.(3.2)}$$

where, $H(T)$ is the entropy of a tree T , I_E is the expected information constant, p_i is the proportion of instances labelled as class i , and J is the total number of classes.

Information gain computes the difference between the information of the dataset (at each node) before and after splitting a specific feature. In other words, it measures the level of expected reduction in uncertainty (impurity or entropy) in the prediction of the sub-tree if the splitting is performed on a given feature [287, 289]. **Eq.(3.3)** is used to compute this measure.

$$IG(T, x) = H(T) - H(T|x) \quad \text{Eq.(3.3)}$$

where T is the tree, x is the feature being evaluated, $IG(T, x)$ is the information gain, $H(T)$ is the entropy of the parent tree and $H(T|x)$ is the weighted sum of the entropy of the sub-trees.

One of the main drawbacks of the purity measure is that it could result in building decision trees that over-fit the dataset as a result of selecting features with a large number of distinct values which have high mutual information. Such features could be the connection number that uniquely identifies connections in the dataset. This criterion is used in the ID3 algorithm [290], which is one of the predecessors of the C5.0.

The **Information gain ratio** aims to address the limitations of the **information gain** metric. It reduces the bias towards features with a large number of distinct values by penalising the selection of a feature based on the number and size of its branches. However, this criterion might result in favouring features with very low information values [291]. This is the criterion used in both the C4.5 algorithm and its improved version, the C5.0 algorithm.

3.2.1.2 Tree pruning

As building decision trees usually over-fits the training data, tree **pruning** is performed. In C5.0, tree pruning is performed by removing parts of the tree that are predicted to have a high error rate [292]. In this pruning process every subtree is evaluated to determine whether it will be replaced with a leaf or a node. There are many factors that influence the pruning process which in turn affect the overall performance of the produced model (tree). Therefore, setting the values of these parameters should be undertaken with care at the tuning phase.

3.2.2 Random Forest (RF)

Random Forest (RF) is basically formed of multiple decision trees that are grown using a combination of ‘Bagging’ and the random selection of features (subspace). Bagging (**B**ootstrap **a**ggregating) is a technique that aims to improve the performance (accuracy and stability) of ML algorithms and to reduce variances and the chances of overfitting [293, 294]. The basic principle behind this technique is to build multiple prediction models and use their aggregated

predictions to produce a final prediction. It works by applying uniform random sampling with a replacement (known as a ‘bootstrap’) from the training dataset to produce a new dataset, which will be used to build a prediction model. The default settings of RF produce a new dataset with the same number of instances as the original training dataset. This sampling is repeated *nTree* times producing as many bootstrap samples. Each of these bootstraps will then be used to build a prediction model, resulting in a total of *nTree* models (decision trees).

After a bootstrap sample is produced, a decision tree is generated. In RF, only a random selection of features (subspace), with no replacement, are evaluated at every node to decide the best split, rather than using the full features set as in the C5.0 algorithm. Usually the number of these random features, *mtry*, is far less than the original number of features.

Out-Of-Bag (OOB) data are the data instances left after a bootstrap sample has been generated. These OOB data are usually used in the internals of RF to estimate and monitor the errors of the decision tree and its strength, as well as the correlation between different trees. Feature importance is also measured using these internal estimates. **Listing 3.1** shows a pseudocode of the Random Forest algorithm [295].

Algorithm: Random Forest (RF)	
Input:	$X = \{(x_1, y_1), \dots, (x_N, y_N)\}$, $nTree = 500$, $mtry = \lfloor \sqrt{p} \rfloor$
Result:	RF model
1	for ($1 \leq i \leq nTree$) do
2	\hat{X}_i = Bootstrap sample from X (Sample N random instances with replacement)
3	$\hat{X}_i^{OOB} = X - \hat{X}_i$ (Out-Of-Bag samples - instances not in the bootstrap sample)
4	Build tree T_i {
5	- Use \hat{X}_i as training data to build the tree,
6	- At every node use randomly selected (with no replacement) <i>mtry</i> features
7	to determine best splits,
8	}
9	Use \hat{X}_i^{OOB} samples to compute internal estimates of tree T_i (errors, strength,
10	correlation, features importance)
11	done

Listing 3.1: Pseudocode of main stages of Random Forest algorithm

The final prediction of the forest is performed by running each instance down all decision trees in the forest. The results of all these trees are then aggregated to form the final decision. For

numerical predictions, the average or the weighted average of the results of all trees is returned, whereas, for classification problems, the majority vote or the probability of the classes is returned. There are many different existing methods in computing these probabilities, such as the proportion of classes returned by the trees or the average of their probability estimates [296]. The latter is the method implemented in the “**Ranger**” package used in all experiments in this thesis [297].

According to Witten *et al.* [298], as cited by Resende and Drummond [299], the RF algorithm has a low training time complexity and fast prediction time. It also has the capability to handle missing data efficiently which means it does not require the data to be pre-processed beforehand, nor does it require the data to be scaled or normalised. Also, due to the bootstrapping feature, the algorithm can handle imbalanced data and rare cases (because of resampling) quite efficiently [300]. It also provides two measures that rate the importance of every feature: the Mean Decrease of Accuracy (MDA); and the Mean Decrease Gini (MDG) [284].

As RF can run more slowly as the number of trees increase, a careful tuning of its main parameters is needed to maintain the required performance. One of the main drawbacks of this algorithm is that its results are difficult to interpret as its model complexities are high. This is due to the number of trees and the randomisation in the sampling of training instances and features [301]. The key stages of the Random Forest algorithm are illustrated in **Figure 3.3**.

3.2.3 Support Vector Machine (SVM)

The Support Vector Machine (SVM) [302] is one of the most popular classification algorithms used for supervised learning tasks in ML. Its development is based on the structural risk minimisation principle [301]. An excellent description of this algorithm can be found in Vapnik’s book [303]. In SVM, each data instance is represented geometrically as a vector (\mathcal{R}^p)

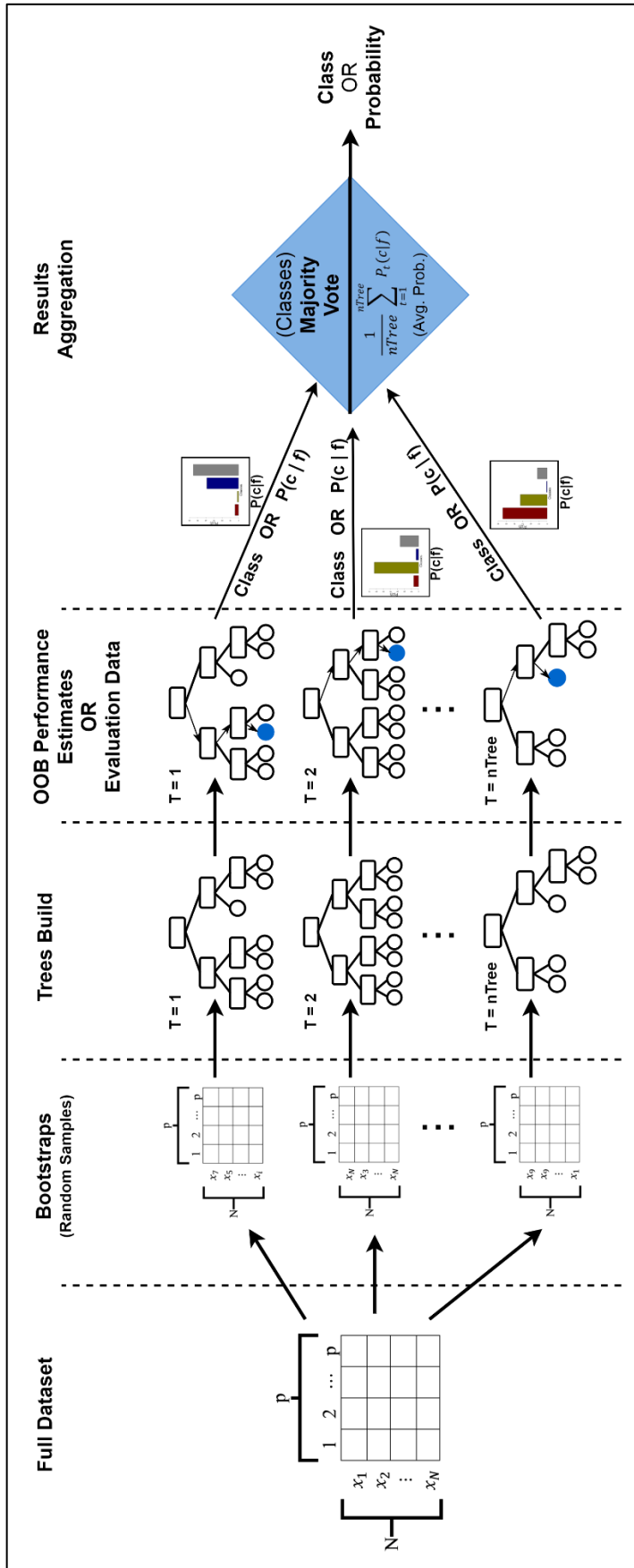


Figure 3.3: Main phases of Random Forest algorithm

in p -dimensional space - $x = (x_1, \dots, x_p) \in X \subset \mathcal{R}^p$. SVM attempts to find a linear surface (hyperplane) - or a line in 2D space - that separates the instances into two classes $y \in \{-1, 1\}$, where this separating hyperplane has the largest distance between the edge points of each class.

These edge points define the border lines for each class as per the following **Eq.(3.4)**:

$$\begin{aligned} \text{Negative } (-) \text{ line equation,} & \quad | w \cdot x + b = -1 \\ \text{Positive } (+) \text{ line equation,} & \quad | w \cdot x + b = +1 \end{aligned} \quad \text{Eq.(3.4)}$$

where, x is an edge point in the training data that lies on the border line of a class, and b is (offset) the distance from the origin to the decision boundary ($w \cdot x + b = 0$) [302]. The edge points also define the width of the margin between those border lines.

These points (vectors) are used to define and outline (support) the separating hyperplane and are called the **support vectors**. The minimum required number of these points is $(p+1)$. For example, in a two-dimensional space, at least 3 data points (vectors) will be the closer to this line and at an equal distance from it, i.e. points a_1 , a_2 and a_3 in **Figure 3.4 (a)**. If any, or all, of these points (support vectors) are removed from the dataset, the separating hyperplane will take a different shape.

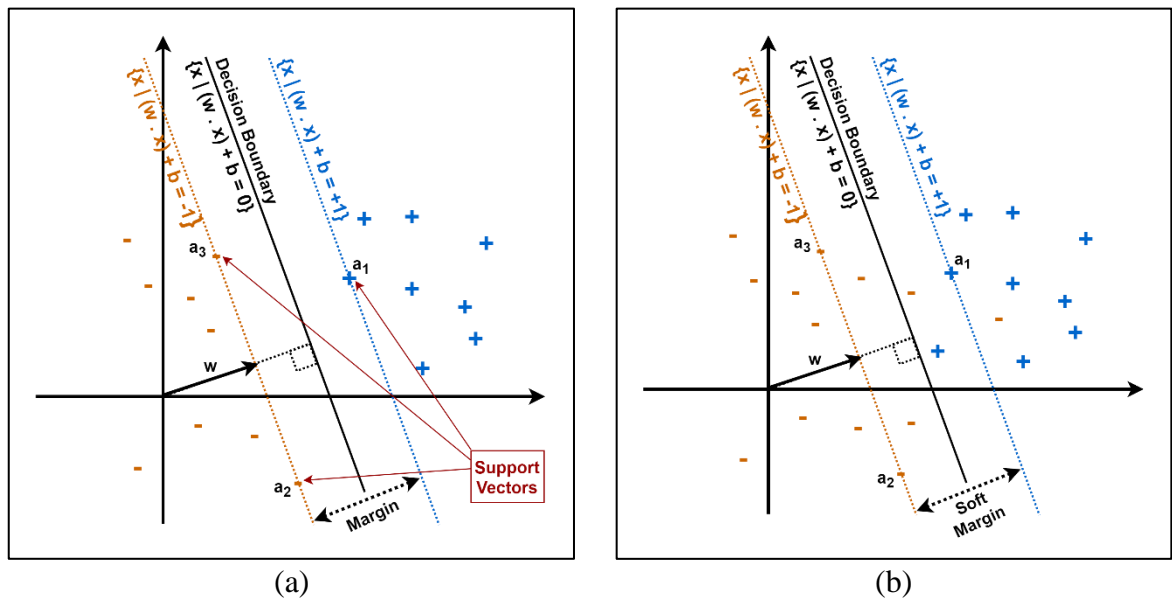


Figure 3.4: Example of SVM on two dimensional dummy data, where a_1, a_2, a_3 : input data points (vectors), w : normal vector to the hyperplane (weight vector) and b : bias. (a) Perfectly separable dataset. (b) Dataset separation with soft margin.

As there could be many separating hyperplanes that might separate positive cases from negative cases, the SVM algorithm searches for a decision boundary with the maximum margin. The width of this margin is the sum of the distances from that decision boundary to the parallel hyperplanes that contain the closest positive and negative training points (support vectors) [304].

The SVM classifier depends on computing w , which is a normal vector perpendicular to the separating hyperplane (decision boundary). This normal vector, precomputed as **Eq.(3.5)** presents:

$$w = \sum_{i=1}^N \lambda_i y_i x_i \quad \text{Eq.(3.5)}$$

where, λ_i are Lagrange multipliers produced at the training phase using data with N training samples. The normal vector is also known as the weights vector [305] which can loosely be thought of as similar to the β coefficients in any regression problems that contain the standardised estimates of variances of dependent and independent variables. Omitting the complex details of calculation and formulas derivation, SVM classification is performed by evaluating which side of the hyperplane a test instance (vector) will fall into, as **Eq.(3.6)** shows:

$$SVM(\hat{x}) = \begin{cases} -1, & w \cdot \hat{x} + b < 0 \\ +1, & w \cdot \hat{x} + b \geq 0 \end{cases} \quad \text{Eq.(3.6)}$$

where, \hat{x} is a test instance, and b is (offset) the distance from the origin to the decision boundary ($w \cdot x + b = 0$) [302] that is precomputed at the training phase.

The classification is usually carried out by checking the sign of the results returned by the above formula for every instance in the testing data. Where the result is greater than 0 (a positive sign) it will be classified as class 1, but where it is less than 0 (a negative sign) it will be classified as class -1.

To address complex datasets where a perfect separating hyperplane does not exist, soft margins are used - see **Figure 3.4 (b)**. For a soft margin, some cost value (C) is passed to the SVM function to allow some violation of the boundary by permitting some levels of mixing between classes. Having a large cost value (C) will increase the margin size which will in turn increase the mixing zone. In this case, the support vectors will not be the ones that are nearest the separating hyperplane, rather they will be on the margin or on the wrong side of the boundary. The tuning parameter (C) has an effect on the variance; small cost values with high variances tend to lead to overfitting, whereas large cost values will reduce variance but show a tendency towards underfitting [306].

One of the main advantages of SVM is that it does not suffer from the “Curse of Dimensionality” as many other ML algorithms do. The “Curse of Dimensionality” refers to the case where the number of observations are far less than the number of features in the dataset. Many ML algorithms tend to suffer from overfitting in this case, which can usually be resolved by a feature reduction process. The ability of SVM to avoid overfitting is linked to its ability to select the right regularisation parameter (or cost value (C)), and the right kernel, due to its carefully tuned parameters for non-linear problems [307]. Also, as SVM errors depend on adjusting the margin which separates the data points in the fitted model rather than the number of features, feature reduction is not required to avoid overfitting as it is in many other ML algorithms.

Some problems will not be linearly separable where SVM cannot produce a good model for such data. These data will therefore require some transformation from input (data) space into higher dimensional (feature) space. The data can be made linearly separable using what is known as kernel methods or functions which can perform such a transformation (**Figure 3.5**).

The resultant separating hyperplane can be expressed using the inner products of the vectors [308]. Some known kernels, like polynomial [309] and Radial Basis Function

(RBF) [310] kernels, are used for such a transformation. Using kernels will incur optimisation costs. In the linear version of SVM, the optimisation process is focused on the tuning parameter (C), but when a kernel is used, all its tuning parameters need to be taken into account [310].

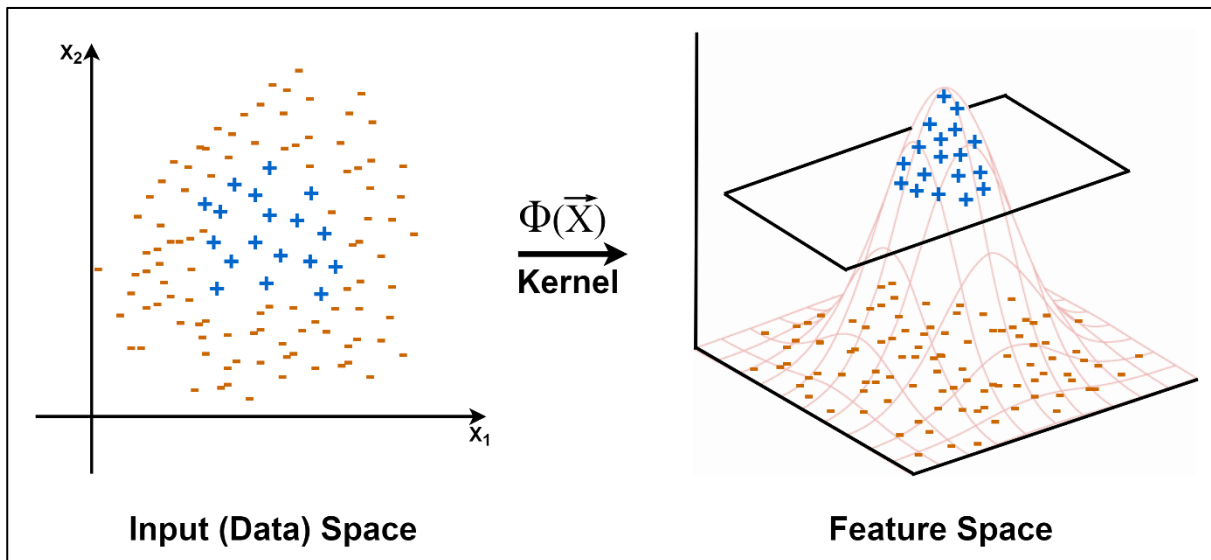


Figure 3.5: Data transformation from 2D in input space to 3D in the feature space using a kernel function. The figure was reproduced with some modification from Statnikov et. al [305]

Some SVM implementations can return the probability of the class. Different implementations will use different techniques to compute such probabilities. For example, Platt [311] proposed a mapping technique to convert the SVM classification into probabilities by using logistic regression. However, this technique has been criticised for not being a true probability and Gaussian processes have been suggested as an alternative [312]. The Platt approach was adopted, with some improvements in the SVM library [313] used for the experiments in this thesis.

SVM processing speed is affected by the kernel used, as some kernels will perform more operations in the transformation phase, which will slow the SVM's speed. For example, the use of a Radial Basis Function (RBF) kernel in a classification problem will run very slowly, as this kernel calculates the distances to all support vectors rather than using a hyperplane equation [314].

SVM's biggest limitation, as noted by Burges [315], is selecting the right kernel. He also points to the size and speed limitations of both training and testing phases. However, a considerable advancement in this area has been achieved as many researchers, such as Lin [316], have developed a number of fast SVM implementations to address this issue. Unlike Random Forest, SVM requires the pre-processing of data to handle cases, such as missing data, and the transformation of categorical data into a suitable numerical format (as SVM only handles numerical data). Transforming (standardizing, normalizing or scaling) the data will have an effect on the results of SVM, and therefore data transformation before training and testing is always recommended.

3.3 Methods Used for Analysis

This section presents the core experimental design employed in all of the experiments undertaken in this thesis. It also sets out the decisions relating to data selection, the preparation requirements, and the parameter settings of the ML algorithms.

3.3.1 Research design

The study performed in this thesis employed a factorial research design and experimental evaluation to assess the performance (i.e. the Geometric Mean of Accuracy) of the models of the three ML algorithms under various setups. The choice of design was driven by the desire to investigate every possible combination, at every level, of all of the factors, on the overall performance of the developed models. In addition, the design was chosen so as to study the relationship between various factors to identify which ML algorithm was the most adaptable to a change in pattern (concept) in the network traffic over time. **Table 3.1** shows the different factors and their levels for every experiment undertaken in this thesis.

3.3.2 Selection of datasets

A number of datasets were used over the course of this research, including synthetic data and simulated domain specific (network flow) data.

Experiments	Factors	Levels
Exp.1	ML Algorithm	3 [C5.0, RF, SVM]
	Thresholds	2 [fixed, adaptive]
Exp.2	ML Algorithm	3 [C5.0, RF, SVM]
	Data Balance	2 [balanced, imbalanced]
	Feature sets	5 [Full, MDA, MDG, MDA _{Bal.} , MDG _{Bal.}]
	Thresholds	2 [fixed, adaptive]
Exp.3	ML Algorithm	3 [C5.0, RF, SVM]
	Data Balance	2 [balanced, imbalanced]
	Feature sets	5 [Full, MDA, MDG, MDA _{Bal.} , MDG _{Bal.}]
	Sample size	11 [10%, 5%, 1%, 0.5%, 0.1%, 0.05%, 0.01%, 0.005%, 0.001%, 0.0005%, 0.0001%]
	Bins (sampling strategy)	5 [B ₁ , B ₁₀ , B ₂₀ , B ₅₀ , B ₁₀₀]
	Errors	4 [0%, 1%, 5%, 10%,]

Table 3.1: Factors and levels of every experiment.

3.3.2.1 Existing Datasets

As the first experiment aimed to investigate the potential of prediction threshold adaptation in addressing the issue of concept drift between training and evaluation datasets, synthetic and domain specific (network flow) datasets were used. The synthetic datasets, SEA [317] and AGR [318], were chosen as they provided the control required over concept drift (between data files) to analyse the effect of threshold adaptation on model predictions when the evaluation data were either of the same or of a different concept to the training data used to build the prediction model. This analysis was extended to the domain specific dataset (gureKDD [160-162]) to investigate the effectiveness of the adaptation approach in a near to real life scenario. The gureKDD was selected over the KDD Cup 1999 and NSL-KDD datasets for two main reasons: firstly, the gureKDD dataset maintains the chronological order of every connection, which makes it possible to split connections based on their time; secondly, the KDD Cup 1999 dataset has a large number of limitations (listed in Al Tobi and Duncan [1]) including miscalculated values, which raised doubts over its suitability.

As the decision to use these synthetic datasets was based on the need to generate different levels of concept differences, the data files generated from each of these datasets had different combinations of similar and differing concepts. Similarly, gureKDD was divided into multiple

files using a time window of one week which resulted in different network patterns (concepts) in the traffic for each week. A detailed discussion on data generation, preparation and descriptions is provided in **Chapter 4**.

3.3.2.2 Newly Generated Dataset

As set out in **Chapter 2**, there are many limitations in the publicly available datasets. In addition, as the aim in this thesis is to extend the analysis to data that were more up to date and more representative of real network traffic, a new dataset was generated by transforming the flows of the ISCX2012 dataset [14] into a suitable format for ML algorithms.

The resultant dataset, called *STA2018*, profiled every connection using 550 features and can be used as balanced or imbalanced (original) data. The decision to transform the ISCX2012 dataset was governed by its availability at the time. It was also the best option out of all of the available datasets. For instance, the ISCX2012 dataset contains modern traffic with full captures. In addition, unlike many other datasets, it provided a reasonable number of simulation days (seven). Although the DARPA datasets have more simulation days than ISCX2012, the traffic included in those datasets is very old, does not represent real network traffic and is very limited. For example, one day of the ISCX2012 captures equals the whole of DARPA in size.

Data records from the resultant dataset (STA2018) were grouped by day so that every data file aggregated all of the connections within that simulation day. Overall, the transformation process had five main stages: *basic-features extraction*; *validation and labelling*; *extending the basic-features*; *balancing*; and *cleaning-up*. Full details of the transformation are discussed in **Chapter 5**.

The large number of features space combined with having a balanced version of the data made it possible to analyse the effect of various decisions such as feature selection and data balancing, which are usually taken at the model development stage. Many of these decisions were based on the overall performance of the prediction models that were developed and tested using the

available training data. This was especially the case when new data with a different concept (and hence, with different important features) were evaluated with the models generated. Therefore, using this dataset made the analysis of the effect of predictions threshold adaptation on these models more systematic.

3.3.3 Parameter setting for the ML algorithms

As stated by Bhuyan *et al.* [12], classification-based methods usually produce better results than unsupervised learning, such as clustering methods, due to the use of data labels at the training stage. However, as shown by Laskov *et al.* [319], the accuracy of the supervised learning algorithms could deteriorate when their models are faced with novel attacks. Therefore, three well-known classification algorithms in the domain of ID were investigated, namely *C5.0*, *Random Forest (RF)* and *Support Vector Machine (SVM)* [see **Section 3.1**]. These algorithms were used to process the training data in order to learn a classification (prediction) model, which was then tested on the evaluation (test) data with different statistical properties (concept). The performance of these models were analysed before and after threshold adaptation to examine their ability to adapt to changing traffic patterns and to detect novel attacks. The following subsections provide further details about the packages used and the parameter settings for each of the algorithms used.

3.3.3.1 C5.0 algorithm

The “c50” package {**version 0.1.0-24**} [320] in the R environment [321] was used in this thesis. All experiments used the default settings of this algorithm, with the 10-trials option (`trials = 10`) set to return the results of the classification as a probability score (`type = "prob"`) when the model was used to predict the evaluation (test) data.

3.3.3.2 Random Forest

The “ranger” package {**version 0.8.0**} [297, 322] in the R environment [321] was used over the course of this research. This package was selected because of its fast implementation of RF in

C++. All experiments used the default settings of 500 trees (`nTree`) to grow, and with the number of features to evaluate at every node being the square root of the total number of features in the dataset (`mtry = $\lfloor \sqrt{p} \rfloor$`), where p is the number of features. The algorithm was instructed to return results in the form of classification probabilities (`probability = TRUE`).

3.3.3.3 Support Vector Machine (SVM)

The open source SVM package (LiblinearR) {**version 2.10-8**} [323, 324] in the R software [321] was used in these experiments. This package executes an optimized linear version of SVM. All experiments used the default settings of *L2-regularized logistic regression* linear model type (`type = 0`) with the cost set to one (`cost = 1`).

The choice to use the linear version of SVM was driven by the very large differences in the runtime of experiments between its linear and nonlinear kernel versions. Some preliminary experimentations have been conducted to compare the two versions. **Table 3.2** presents the runtimes (in *seconds*) of the kernel SVM (with `type = 2` [radial basis function], `cost = 2` and `gamma = 2`) and the linear SVM (with `cost = 2`). These experiments were performed on 10% subsets of Day 2 (12/Jun) and Day 3 (13/Jun) of the STA2018 dataset. This table shows the large difference between the two versions where the linear version of SVM is much faster. It also shows that the runtime of the kernel SVM grows exponentially as the number of instances increase, especially when the data is balanced.

As SVM can only handle numerical data, it was necessary to pre-process the data before the training or testing phase took place. Therefore, all categorical (nominal) features were converted into dummy attributes, so that every value for each of the levels of these categorical features was converted into an independent feature that contained a 'zero' or a 'one' [325].

Another pre-processing stage, as recommended by the SVM library package [323, 326], was to standardise the data. Therefore, every feature in the training data was standardised (as per

	Day 2 (12/Jun) 10%				Day 3 (13/Jun) 10%			
	Kernel Cost=2 , Gama=2		Linear Cost=2		Kernel Cost=2 , Gama=2		Linear Cost=2	
	Time Sec. (imbalanced)	Time Sec. (balanced)	Time Sec. (imbalanced)	Time Sec. (balanced)	Time Sec. (imbalanced)	Time Sec. (balanced)	Time Sec. (imbalanced)	Time Sec. (balanced)
Fold 1	3,330	16,602	4	4	4,186	23,693	4	6
Fold 2	3,505	12,101	2	4	3,463	21,595	4	4
Fold 3	3,385	12,309	2	6	4,054	20,559	2	6
Total (seconds)	10,220	41,012	8	14	11,703	65,847	10	16

Table 3.2: Runtime in seconds between kernel and linear SVM setups on a 10% subset of Day 2 (12/Jun) and Day 3 (13/Jun) of the STA2018 dataset

Eq.(3.7)). The standardisation parameters (the mean and standard deviation) of the training data, which is utilized to build the classification model, were used to standardise the features of the test data before being classified by the model:

$$\dot{x} = \frac{x - \mu}{\sigma} \quad \text{Eq.(3.7)}$$

where, \dot{x} is the new standardised value, x is the actual value to be standardised, μ is the mean and σ is the standard deviation of a feature column.

In addition, all class labels were mapped for every dataset as shown in **Table 3.3**.

Dataset	Class label	SVM label
SEA and AGR	groupA	-1
	groupB	+1
gureKDD and STA2018	Attack	-1
	Normal	+1

Table 3.3: Class labels mapping for SVM algorithm.

3.3.4 Evaluation measures

As discussed in **Chapter 2**, many of the common evaluation measures suffer in model performance assessment when imbalanced data are used. It has also been shown that network traffic exhibits an imbalance of traffic classes i.e. more normal traffic exists than anomalous traffic [98]. In an attempt to avoid this problem, the **Geometric Mean (G-Mean) of Accuracy** [231] metric (see **Section 2.3.2.4**) was adopted throughout this thesis. This measure was used

as the main measure to evaluate the performance of all of the models developed in all experiments in this thesis.

3.3.5 Performance assessment techniques

Cross-Validation is a technique used to estimate a model's performance and to assess how its results could be generalised for unseen datasets [327-329]. This technique addresses the issue of data shortage when the holdout technique (for training and testing sets) loses important modelling or testing capability [330]. It can also be used to assess the expected performance of a prediction model in a real environment. In addition, it is a very useful technique to avoid **overfitting** problems, the situation that arises when the model perfectly predicts the data used to generate the classification model, but fails to generalise to new data. It is also a useful technique to identify and fine tune the main parameters of the model [331]. As discussed in **Section 2.3.1**, there are many types of Cross-Validation techniques, ranging from 'leave-one-out' to 'holdout' methods, but the most popular one is the K-folds Cross-Validation. In K-folds Cross-Validation (**Figure 3.6**), the dataset is randomly divided into K parts. A model is then trained using K-1 parts and tested on the remaining part. This process is repeated K times, so that each one of the K parts is only used once as test data. The model's overall performance is estimated by aggregating the performance of the K models (through averaging or a majority vote). Although this evaluation method is frequently cited in the literature, it requires a long time to process as larger values of K are used. Also, as discussed in **Chapter 2**, it could provide overly optimistic results due to the random division of datasets.

Despite these drawbacks, the K-folds Cross-Validation technique was used in all experiments at every model building (training) stage to estimate the prediction thresholds for every developed model as per the recommendation of Ambroise and McLachlan [205].

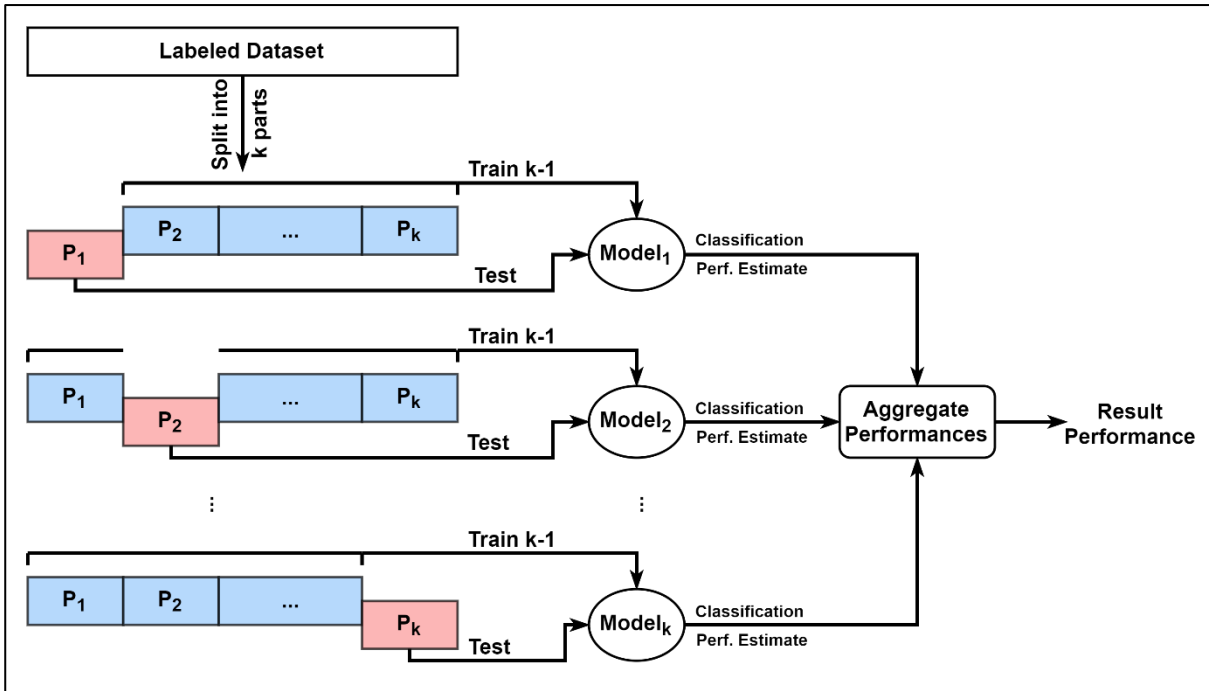


Figure 3.6: K-folds Cross-Validation process.

Prospective sampling [225], as discussed in **Section 2.3.1**, is a technique underused in batch-learning tasks in the domain of network ID. However, this was the sampling approach used in this thesis to evaluate the performance of the generated models, by assessing them using evaluation (test) data collected at a different time from the training data. This evaluation method aimed to mirror real life, given that models are usually trained on data that have been collected in the past to predict future data.

3.3.6 Statistical evaluation

Normality tests were used to assess whether or not the data was following a normal distribution. Such tests are useful in selecting the right statistical analysis tests, such as parametric or non-parametric tests. As a result, in order to assess the normality of the results prior to analysis, the **Shapiro-Wilk** normality test [332] was used. However, this test is limited in its ability to handle more than 5,000 records. Therefore, in such cases (as in Experiment 3 and discussed in **Chapter 7**) the **Anderson-Darling** normality test [333, 334] was used instead.

Friedman's test [335, 336] is a non-parametric test used to assess the difference between treatments or effects during multiple test trials, without assuming a specific distribution in the data. This test involves ranking the observations in each row, then computing the mean of the ranked observations across treatments/effects (for each column).

Parametric tests, such as ANOVA, require asserting some assumptions before they can be applied. One of these assumptions is that the data should be normally distributed. As this core assumption could not be made for these experiments, a non-parametric approach was applied to analyse the results. Using an example, the following figure (**Figure 3.7**) illustrates how the Friedman test works.

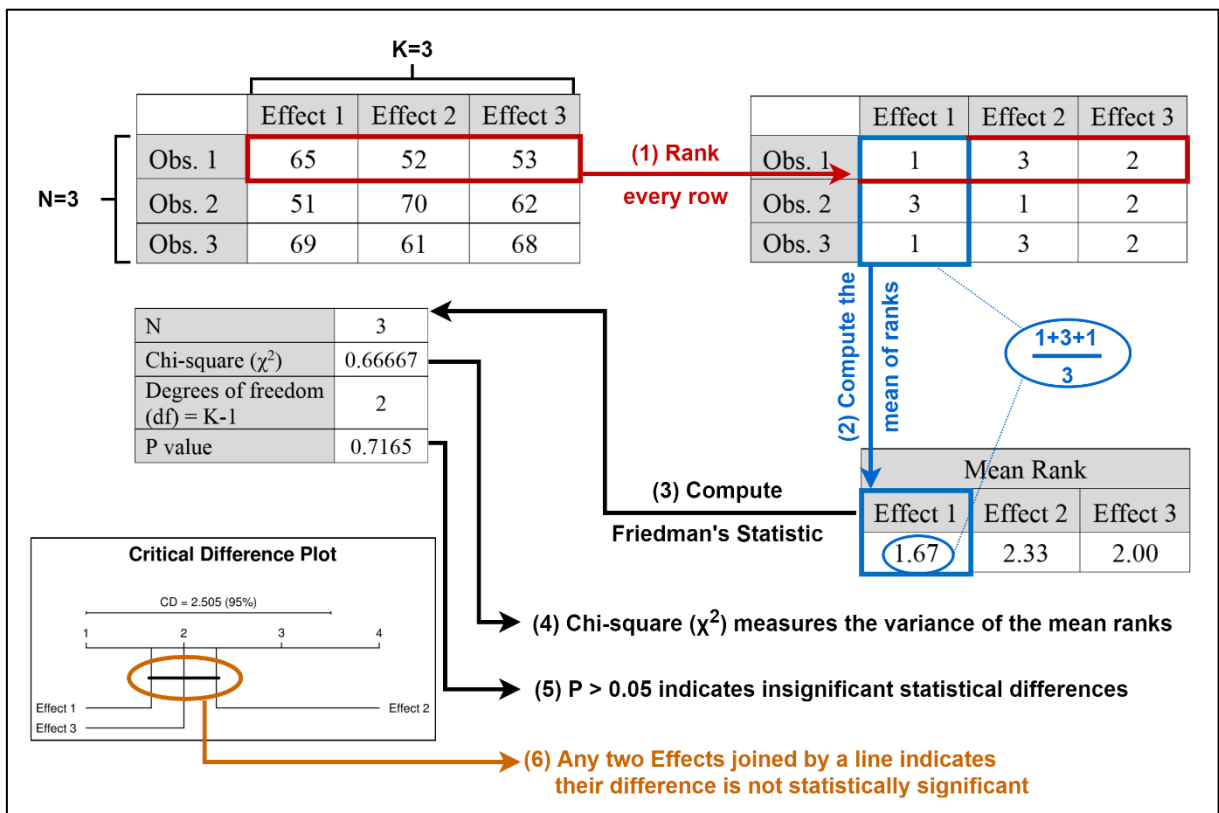


Figure 3.7: Friedman's Test computation and interpretation

If the Friedman test indicates that there is a significant difference between treatments/effects, post-hoc tests [337] can then determine which of these treatments/effects are significantly different by comparing the differences between their mean ranks. In these experiments, the **Nemenyi post-hoc test** [338] was used to calculate pairwise comparisons for different

treatments and to determine which pairs' differences were statistically significant. To visualise these differences Demšar [17] proposed the *Critical Difference Plot*; in this plot, any two effects joined by a line indicates that they are not statistically different (see **Figure 3.7**).

3.4 Limitations

There are a number of unavoidable limitations introduced in this thesis related to many factors, such as the data selection and the adopted research design. Firstly, the experiments only focus on synthetic data and simulated network traffic. No real network traffic was used. This is because high variability exists in real traffic and the true state of the traffic would be unknown which would render the analysis unreliable due to the lack of control required over various variables and parameters.

Secondly, the analyses only measured a model's performance by evaluating its accuracy i.e. the Geometric Mean of Accuracy, so no speed or resource utilisations were considered. This was due to the fact that the core aim of the thesis was to investigate the effect of threshold adaptation on model performance i.e. accuracy. Other efficiency factors were considered engineering issues and out of scope.

Thirdly, the choice of the factorial research design limited the number of ML algorithms that could be compared. As the factorial design could have resulted in an exponential growth in the number of performed experiments and analyses required, it was decided to limit the number of ML algorithms to the three most widely known ones.

Finally, although the main drawback of the factorial experiment design was the large number of treatment-level combinations which would then require a similar number of experimental units (records), this drawback did not affect the experiments conducted as all combinations were represented. However, with such a high number of factor level combinations, analysis became difficult and necessitated a complicated interpretation of the results. In addition, more

time was needed than anticipated to undertake the experiments to address all of the combinations.

3.5 Summary

As noted above, the aim of this thesis is to create a better understanding of the interplay between discriminative threshold adaption of model predictions and the overall performance of a model (its Geometric Mean of Accuracy) to predict anomalies in local network traffic with different levels of concept or feature drifts. A further aim was to demonstrate that threshold adaptation for evaluation/test data can be achieved through optimising a performance measure based on validation data that is randomly sampled from and representative of the whole population (evaluation/test data). An empirical approach was therefore used to investigate the use of fixed threshold for model predictions and assess whether they undermine the real prediction power of a detection model. This investigation studied the effect of threshold tuning on prediction models developed with different ML algorithms and various setup scenarios. As such, three main questions formed the basis of the research:

- How will the detection accuracy of an adaptive discriminating threshold of the predictions of a batch binary-based anomaly ID model compare to the accuracy of a fixed threshold?
- Can the adaptation of the discriminating threshold improve the accuracy of a binary-based anomaly ID model when evaluated network traffic has different salient features than those used to build the predictive model?
- Can the optimal discriminating threshold be identified using a labelled small sample of the evaluated network traffic under the batch-learning setup?

The remainder of this thesis addresses these questions. The study consists of an incremental empirical investigation, where the findings of one question are used as an input to the next.



Chapter Four

Adaptive Cutoff (Threshold) for Prediction Models

This chapter provides an analysis of binary (attack or normal) batch-based classification problems using a prospective sampling approach. In this approach, each experimental dataset is divided into subsets (batches) whereby each one is used to generate a binary prediction model to be tested on other subsets. This experiment aims to investigate the extent to which a detection model can be utilised to classify previously unseen data using only small parts of the data to build the learning model. It also aims to shed light on a serious limitation in assessing model prediction performance i.e. that a model's performance is highly related to the evaluation data used to assess the model and should not be generalised as a single measure for the model. Also, model performance on evaluation data should be analysed separately and not on a pre-set threshold which was determined using some validation dataset or some form of Cross-Validation technique.

4.1 Problem Statement

A model generation process usually requires as many examples as possible to produce accurate models. For a model to perform well, training and evaluation datasets should exhibit similar statistical properties. As statistical similarity changes over time (known as concept drift [138, 140, 141, 339, 340]), the predictive power of these models should become less accurate. Therefore, the common practice in batch predictive analytics requires the use of large datasets

in model generation. Such practice aims for a dataset that would manifest as many statistical properties of future unseen data as possible. In practice, this requirement is difficult to satisfy, especially in dynamic environments, such as *inter alia* network traffic, the stock market, self-driving vehicles or spam filters.

Many solutions have attempted to address this issue by devising tools and techniques to learn from an evolving data stream [135, 279, 341]. Although this area of research is starting to produce potential solutions, it is beyond the scope of this thesis which is focused on binary batch learning.

All proposed methods using predictive analytics and ML for (batch or stream) classification problems use the same basic principle in tackling the problem by using a vector of variables $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_p)$ of an unlabelled record to assign it to one of the predetermined classes $\mathbf{y} \in \{\mathbf{c}_1, \dots, \mathbf{c}_n\}$ [341].

This chapter evaluates the extent to which a detection model that was trained on a small subset can predict larger subsets of the same domain/problem, for example, in network domains where a model is developed using one day/week for network traffic to predict future days/weeks. The evaluation in this chapter compares the **G-Mean Accuracy** and the **Area Under the ROC Curve (AUC)** of these models due to their insensitivity to data imbalance.

4.2 Proposed Solution

This chapter investigates an approach that attempts to address a real life setup, where the training data (known as labelled data) are much smaller than the evaluation data (unlabelled data). It analyses the effect of this approach by adapting the classification results of a binary predictive model on evaluation data in order to maximise its performance on real data as well as to achieve a more accurate and reliable reading of that model's performance. This will give the model producers (system analysts) the ability to accurately determine when a prediction

model is no longer valid and when model updating is required. This approach uses the **prospective sampling** technique to evaluate models [225] (see **Section 2.3.1**), where a predictive model is evaluated using data collected at a different time to that of the training data, such as a day/week.

The experiments to test the potential of this approach, outlined in this chapter, had two stages. In the **first** stage, the well-known 10-folds Cross-Validation technique was performed on aggregated parts of every dataset's files using different classification algorithms. This stage aimed to identify the algorithms' optimal performance on different datasets when training and testing data exhibit the same statistical properties as data that are randomly sampled from the same (aggregated) population. Such models should be expected to achieve the highest performance measures. This stage aims at evaluating the following hypothesis: *“there is no statistically significant difference in model performances (G-Mean accuracies) between the different algorithms”*.

In the **second** stage, each subset of a dataset was used to generate a prediction model (using the same classification algorithms). The 10-folds Cross-Validation technique was used at the model generation to compute the optimal discriminating threshold for the overall predictive model. Every generated model was tested on the remaining subsets separately within the dataset. Two performance readings of every model were recorded at this stage using the same measure (G-Mean Accuracy); one was based on the original model's threshold computed at the generation phase, and the second, on its performance after threshold adaptation on the evaluation (test) subset. This stage aimed to depict real life situations where training and testing datasets have different statistical properties. As a result the following hypothesis is to be tested: *“there is no statistically significant difference in model performances (G-Mean accuracies) before and after cutoff (threshold) adaptation between the different algorithms”*.

The threshold adaptation is performed to find the cutoff point at which the **G-Mean Accuracy** is the maximum. **Listing 4.1** presents the computation process used in finding the adapted threshold for every evaluated dataset based on the prediction scores returned by the prediction models and its known labels.

Algorithm: Threshold Adaptation	
Input:	preds=Model predictions of the data , labels=data labels
Result:	thr _{adpt.} =The adapted threshold
1	preds <- sort(preds) // sort the predictions list of the data in an ascending order
2	labels <- sort(labels) // sort the labels list of the data as their predictions' order
3	
4	thr _{adpt.} <- 0
5	gAcc.best <- 0
6	
7	for prd in preds do
8	fp <- getFP(preds, labels, prd) // get the FALSE POSITIVE at threshold prd
9	tp <- getTP(preds, labels, prd) // get the TRUE POSITIVE at threshold prd
10	fn <- getFN(preds, labels, prd) // get the FALSE NEGATIVE at threshold prd
11	tn <- getTN(preds, labels, prd) // get the TRUE NEGATIVE at threshold prd
12	
13	tpr <- (tp / (tp+fn))
14	tnr <- (tn / (tn+fp))
15	
16	gAcc <- sqrt(tpr * tnr)
17	
18	if(gAcc > gAcc.best){
19	thr _{adpt.} <- prd // set the adapted threshold at the point gAcc is maximum
20	gAcc.best <- gAcc
21	}
22	
23	done
24	
25	return(thr _{adpt.}), // Return the adapted threshold

Listing 4.1: Pseudocode of threshold adaptation process and the selections of the optimal threshold for the evaluated data

Further details about the experiments' setups and configurations are presented in **Section 4.4**. Every generated model was trained for binary classification and configured to return the probability of the class rather than the class label. An advantage of using class probability was the flexibility it offered in computing a model's performance at different prediction thresholds and in determining the point of maximum performance.

4.3 Datasets

This section provides an overview of the datasets used in the experiments outlined in this chapter. Two synthetic datasets (SEA and AGR) [317, 318] were generated randomly,

alongside one domain specific dataset (gureKDD) [160-162]. The latter is a transformation of the network traffic of the DARPA 1998 dataset which is similar to KDD 1999 but much cleaner.

4.3.1 *gureKDDcup*

gureKddcup [160-162] (referred throughout this chapter as gureKDD) is a transformation of the raw network traffic of the DARPA 1998 dataset [342] into a suitable format for ML tasks, where every connection is described using a set of features. This transformation is similar to the KDD 1999 dataset [156] but much richer. All connections in this dataset can be linked back to their origin in DARPA traces and every connection has a unique ID that helps identify the chronological order of all connections. Traffic payloads are available in separate files labelled by connection ID. Therefore, all connections in this dataset are chronologically separable and can be divided by day, week, *etc.*

For these experiments, all traffic (over a seven week period) was segregated into a time window of a week, which resulted in seven files. Every file contained the network traffic of that week (Monday-Friday). Every connection in these files was profiled using 41 features; 3 of which were nominal (*protocol_type*, *service* and *flag*), 6 were binary features, 15 were continuous (real) features, and 17 were integer features. These features were divided into four main groups: intrinsic (basic) features [1-9]; content-based features [10-22]; time-based features [23-31]; and connection-based features [32-41].

Each connection was labelled as either **normal** or as one of the 35 different attacks. These attacks were grouped into four main classes: **DOS**, **Probing**, **Remote-to-Local** or **User-to-Root**. In these experiments, the data were pre-processed so all different attack types were grouped and labelled as ‘**attack**’ to produce binary classes. **Table 4.1** presents a statistical summary of the connection class types for each of the seven weeks.

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Total
Normal	177,889	186,706	72,676	98,627	128,516	247,699	217,743	1,129,856
Attack	21	2,084	215,693	15,319	475,787	703,662	217,072	1,629,638
<i>DOS</i>	16	1,002	207,896	1,171	465,825	684,741	207,035	1,567,686
<i>PROBE</i>	0	1,027	7,757	12,366	9,941	18,017	10,031	59,139
<i>R2L</i>	1	55	39	1,752	0	881	2	2,730
<i>U2R</i>	4	0	1	30	21	14	4	74
<i>Anomaly</i>	0	0	0	0	0	9	0	9
Total	177,910	188,790	288,369	113,946	604,303	951,361	434,815	2,759,494

Table 4.1: Number of connection classes in every file in the gureKDD dataset

Table 4.1 shows the number of normal and attack connections for each week or file and also presents the breakdown of attack by class for each data file. It clearly shows a different class balance in each week, For example, Week 1 (file 1) was the worst as it contains only 21 (0.0118%) attacks.

4.3.2 SEA

A Streaming Ensemble Algorithm (SEA) generator [317] in the MOA framework [143] was used to generate a data stream with three continuous features (X_1 , X_2 , X_3). Each feature had a range between 0 and 10, although only features, X_1 and X_2 , influence the class value. Instances were produced by randomly generating points (X_1 , X_2) in a two dimensional space. Instances were labelled as *groupA*, if $X_1+X_2 > \theta$, and as *groupB*, if $X_1+X_2 \leq \theta$, where X_1 and X_2 were the first two features and θ was a threshold. There were four functions which would label the instances differently based on their threshold values between the two classes (function 1 sets $\theta=8$, function 2 uses $\theta=9$, function 3 sets $\theta=7$, and function 4 sets $\theta=9.5$) [343]. The SEA generator's default setting was used to add 10% noise classes. Six different data streams (files) were produced: function 1 was used to generate two streams (file 1 and file 2); function 2 was used to generate two other streams (file 3 and file 4); and a combination of function 1 and function 2 was used to generate two streams (file 5 and file 6). For every file, calls to these functions used different seed values to set the seed of the random generator function to generate

new random instances. **Figure 4.1** presents an example of the command line call to generate File 1 with the SEA stream generator.

```

java -cp moa.jar -javaagent:sizeofag-1.0.0.jar moa.DoTask
  "WriteStreamToARFFFile -m 200000 -f f1.arff -s (generators.SEAGenerator -f 1 -i 1)"

WriteStreamToARFFFile Parameters:
  -m : "Maximum number of instances to write to file."
  -f : "Destination ARFF file name."
  -s : "Stream to write."

generators.SEAGenerator Parameters:
  -f : "Classification function used to assign instances with class labels."
  -i : "Seed for random generation of instances."

```

Figure 4.1: Command used to generate File 1 of SEA dataset.

Each stream consisted of 200,000 instances. This dataset was used to analyse the effect of different statistical properties (concept drift) between training and testing data on the model's performance. **Table 4.2** lists the number of instances of each class in every file in this dataset.

	File 1	File 2	File 3	File 4	File 5	File 6	Total
groupA	71,609	71,298	85,190	84,965	78,295	77,913	469,270
groupB	128,391	128,702	114,810	115,035	121,705	122,087	730,730

Table 4.2: Number of instances' classes in every file in the SEA dataset

4.3.3 AGR

The AGRRAWAL generator [318] in the MOA framework [143] was used to generate a data stream with nine features (X_1, \dots, X_9), six of which were nominal (factor) and three of which were continuous. This generator had ten different functions to assign the produced instances to one of two different classes, based on the values of their different features. The following examples illustrate the labelling rules of the two functions that were used in generating this dataset:

Function 1: - if (age < 40 OR age \geq 60) then *groupA* else *groupB* ,

Function 2: - if (age < 40){ if (50K \leq salary \leq 100K) then *groupA* else *groupB* },
 - else if (age < 60){ if (75K \leq salary \leq 125K) then *groupA* else *groupB* },
 - else{ if (25K \leq salary \leq 75K) then *groupA* else *groupB* } ,

Each function increases the level of complexity as it uses additional features and complex rules to label the instances [344]. The generator's default setting was used to add 10% noise classes by introducing a disturbance factor that added a deviation value (following uniform random distribution) to the original feature's values. Six data streams were produced: Function 1 was used to generate two streams (File 1 and File 2); Function 2 was used to generate two other streams (File 3 and File 4); and a combination of function 1 and function 2 was used to generate two more streams (File 5 and File 6). All function calls used different seeds to randomly generate different instances for each file. Each stream consisted of 200,000 instances. **Figure 4.2** provides an example of the command line used to generate the data of File 1 in this dataset.

```

java -cp moa.jar -javaagent:sizeofag-1.0.0.jar moa.DoTask
  "WriteStreamToARFFFile -m 200000 -f f1.arff -s (generators.AgrawalGenerator -f 1 -i 1)"

WriteStreamToARFFFile Parameters:
-m : "Maximum number of instances to write to file."
-f : "Destination ARFF file name."
-s : "Stream to write."

generators.AgrawalGenerator Parameters:
-f : "Classification function used to assign instances with class labels."
-i : "Seed for random generation of instances."

```

Figure 4.2: Command used to generate File 1 of AGR dataset.

Table 4.3 presents a summary of the labels frequency in every file for this dataset.

	File 1	File 2	File 3	File 4	File 5	File 6	Total
groupA	134,572	134,457	76,577	76,947	105,301	105,785	633,639
groupB	65,428	65,543	123,423	123,053	94,699	94,215	566,361

Table 4.3: Number of instances' classes in every file in the AGR dataset

4.4 Experimental Setting

The experiments discussed in this chapter have been evaluated, in terms of classification performance, using **G-mean Accuracy** and **AUC**. These experiments were executed in two different phases as explained below and illustrated in **Figure 4.3**.

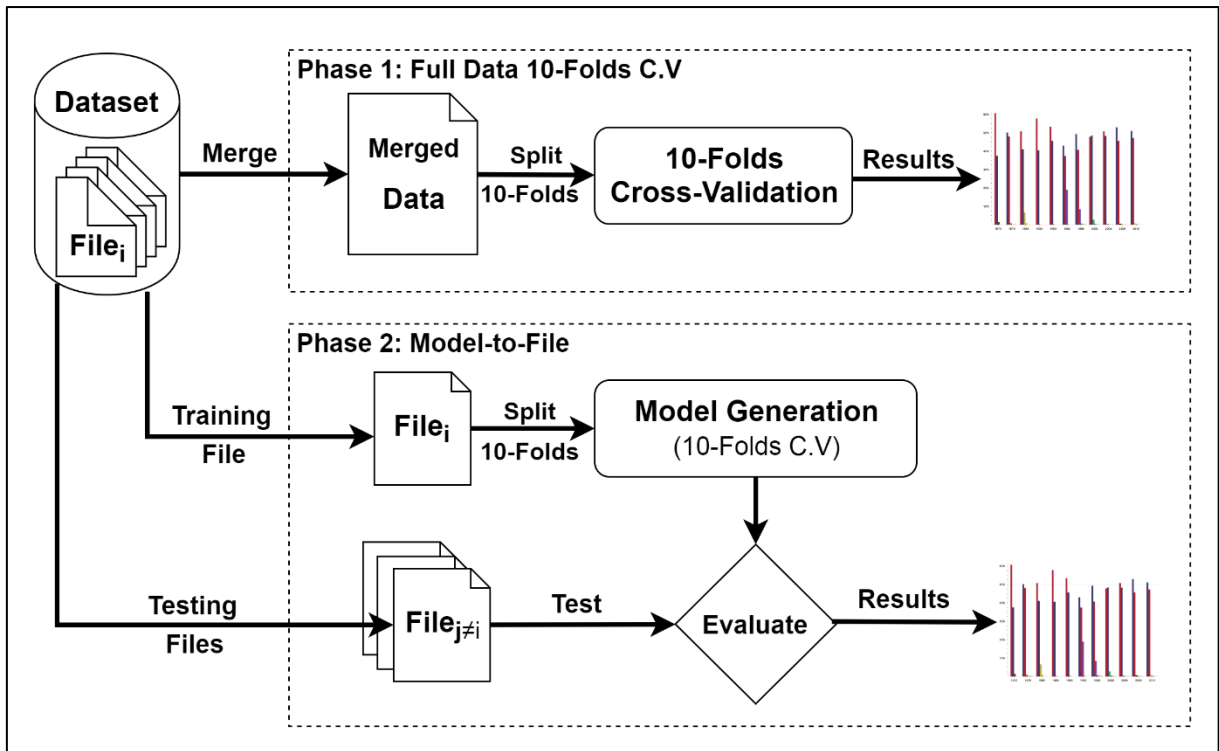


Figure 4.3: The experiments' phases diagram.

In the first phase, all the files in each dataset (gureKDD, SEA and AGR) were merged to form the total population of the dataset. The merged dataset was then used to produce binary classification (prediction) models using three different algorithms (C5.0, Random Forest and SVM). Models were assessed using 10-folds Cross-Validation. This stage looked at the conventional method of model development, where training and testing data display the same statistical properties as they are randomly sampled from the same population. The model's final prediction threshold (optimal cutoff) was computed by aggregating all the fold's predictions and finding the point at which maximum **G-mean Accuracy** was reached. An experiment for every combination of dataset and algorithm was repeated ten times.

In the second stage, every file (subset) in the dataset problem was used to generate a single model, and this model was tested on the remaining files (subsets) within the dataset. The aim of this strategy was to address the classification performance of the generated models where training and testing data had different statistical properties.

To identify any statistically significant differences between the models at each phase, the Friedman test was used with significance level $\alpha=0.05$, to assess the null hypothesis, “*there were no statistical differences between models*”.

Experiments were performed on a “Dell C5220 PowerEdge Rack Servers” cluster, which had 12 micro servers. Each micro server has dual quad-core Intel Xeon 3.4GHz CPUs, 16GB RAM, two 500GB SATA disks, and two Gigabit Ethernet interfaces.

4.5 Results and Discussion

These experiments started out by comparing the detection performances of three different, well-known algorithms in ML (C5.0, Random Forest and SVM) on three different datasets (gureKDDcup, SEA and AGR). In this set of experiments the conventional method of 10-folds Cross-Validation technique was applied to the merged files of each dataset, where the maximum **G-mean Accuracies** of these models and the best cutoff values had been reported. Each experiment was repeated ten times. As the results showed only a minimal variability (see **Table 4.4**) there was no need to do more repetitions.

In the second set of experiments, the same datasets and algorithms were used to generate detection models but in scenarios that were similar to natural settings (prospective sampling technique). In these experiments, models were generated on a subset of the dataset using the 10-folds Cross-Validation technique to set these models’ parameters i.e. the cutoff. These models were then used to evaluate the remaining files in the dataset. Two G-mean accuracy values were computed for every combination of prediction model and evaluation data. The first G-mean accuracy was obtained when the model’s pre-set cutoff value, which was calculated using the 10-folds Cross-Validation, was used to predict the data file. The second G-mean accuracy value was calculated based on the maximum accuracy reached when the prediction cutoff value was adapted to the evaluated data file.

4.5.1 10-folds Cross-validation on Full Data

This section presents the results from using the full data for each dataset (gureKDD, SEA and AGR) in model generation using 10-folds Cross-Validation on the three algorithms (C5.0, Random Forest and SVM).

Table 4.4 presents the mean performances of the ten trials of the 10-folds Cross-Validation in terms of the G-Mean Accuracy and the Area Under the ROC Curve (AUC) of the three algorithms (C5.0, Random Forest and SVM). It also shows the mean of the optimal cutoff values of the ten runs at which maximum G-Mean Accuracies were reached.

In general, all algorithms should reach similar accuracies for their respective datasets. However, in the artificial dataset AGR, SVM failed to perform anywhere close to C5.0 or Random Forest (showing a difference of almost 15% - see **Table 4.4**). This could have been down to the nature of the dataset, which could be non-linearly separable as a linear version of SVM was used in this analysis. Further investigation would have been needed to analyse the effect of data transformation using some kernel functions on non-linear versions of the SVM implementation. However, that would have been beyond the scope of this experiment. In general, Random Forest is capable of improving detection performance on all datasets.

Generally, the performance of all algorithms on gureKDD was the highest, followed by those on the SEA dataset. The AGR dataset was the worst in reaching high detection accuracy. This fact is clearly illustrated by the plots in **Figure 4.4** which show the G-Mean Accuracy curve against the cutoff values for all datasets. These plots show the ten runs in a lighter colour and the mean of these runs in solid colour. They also show the optimal cutoff values for each dataset under the tested algorithm.

	C5.0			Random Forest			SVM		
	AUC	G-Mean Accuracy	Optimal Cutoff	AUC	G-Mean Accuracy	Optimal Cutoff	AUC	G-Mean Accuracy	Optimal Cutoff
gureKDD	0.9999 ±0.0	0.9998 ±0.0	0.5322 ±0.0122	0.9999 ±0.0	0.9998 ±0.0	0.47143 ±0.0126	0.9991 ±0.0	0.9947 ±0.0	0.5879 ±0.0022
SEA	0.8851 ±0.0001	0.8568 ±0.0002	0.2959 ±0.0070	0.94529 ±0.0001	0.8951 ±0.0002	0.23289 ±0.0011	0.8856 ±0.0	0.8621 ±0.0	0.4354 ±0.0001
AGR	0.832 ±0.0001	0.7162 ±0.0003	0.5322 ±0.0044	0.7103 ±0.0001	0.65798 ±0.0001	0.77004 ±0.0011	0.5621 ±0.0	0.5627 ±0.0	0.5144 ±0.0002

Table 4.4: Average model performances (AUC and G-Mean Accuracy), the average Optimal Cutoff value (at which maximum G-Mean Accuracy was reached) and their standard deviation of the 10-folds Cross-Validation (10 repetitions)

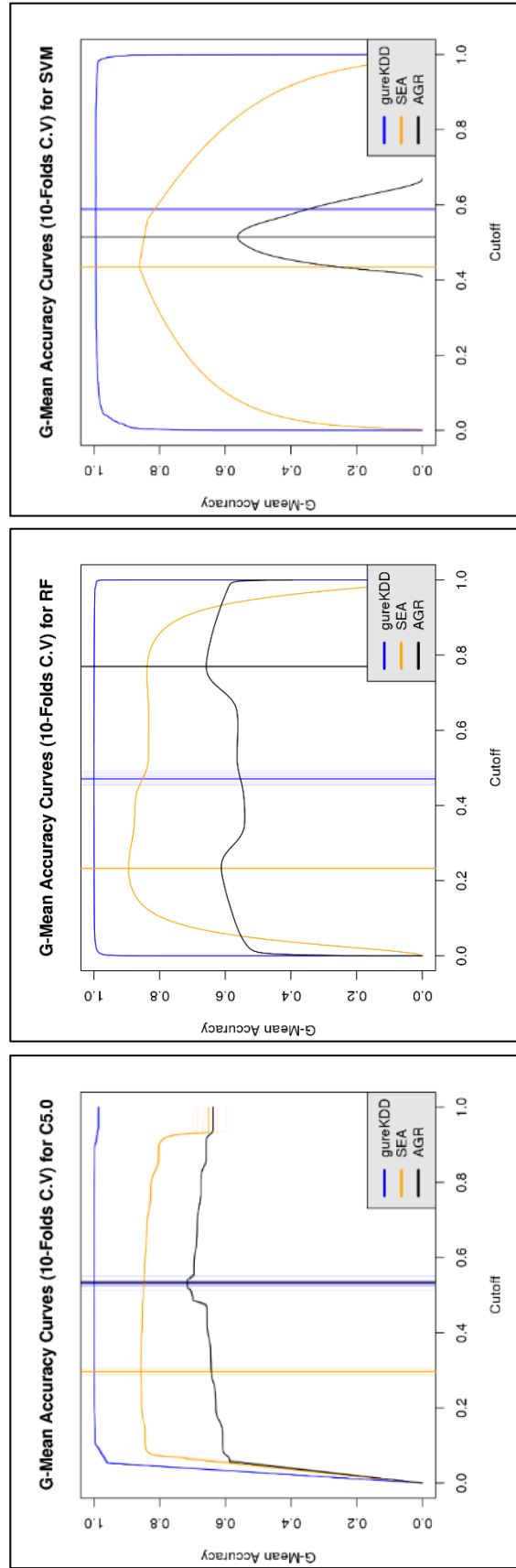


Figure 4.4: G-Mean accuracy curves of the 10 runs of the 10-folds Cross-Validation experiments for the three datasets (gureKDD, SEA and AGR) using three classification algorithms. (a) C5.0. (b) Random Forest. (c) SVM.

The performance of every run of the 10-folds Cross-Validation on the full datasets was rearranged into a matrix and analysed using the Friedman test to evaluate any significant differences between the algorithms used. The non-shaded part of **Table 4.5** sets out the model performances of every run as a matrix.

Dataset	Run	C5.0	SVM	RF
gureKDD	1	⋮	⋮	⋮
	⋮	⋮	⋮	⋮
	10	⋮	⋮	⋮
SEA	1	⋮	⋮	⋮
	⋮	⋮	⋮	⋮
	10	⋮	⋮	⋮
AGR	1	⋮	⋮	⋮
	⋮	⋮	⋮	⋮
	10	⋮	⋮	⋮

Table 4.5: Model G-Mean accuracies arrangements of the 10-folds Cross-Validation on the full dataset for Friedman's test.

Friedman's test was used to analyse whether the difference between these algorithms was significant. The tested hypothesis was, "*there is no statistically significant difference in model performances (G-Mean accuracies) between the different algorithms*". This test revealed that there was a significant difference between the different algorithms applied to these datasets under the 10-folds Cross-Validation approaches, $\chi^2(2) = 26.7$, $p = 0.000 < 0.05$. The follow up Nemenyi post-hoc test revealed that the algorithms were all different from each other, as illustrated in **Table 4.6**, which shows that the p-values of all pairwise comparisons, were less than 0.05.

	C5.0	SVM
SVM	0.027	-
RF	0.027	0.000

Table 4.6: Results of the pairwise Nemenyi comparison test for the full datasets 10-folds Cross-Validation experiment.

Figure 4.5 presents these results in a graph, shows that no two algorithms were joined by a line, which indicates that the differences between the algorithms were statistically significant.

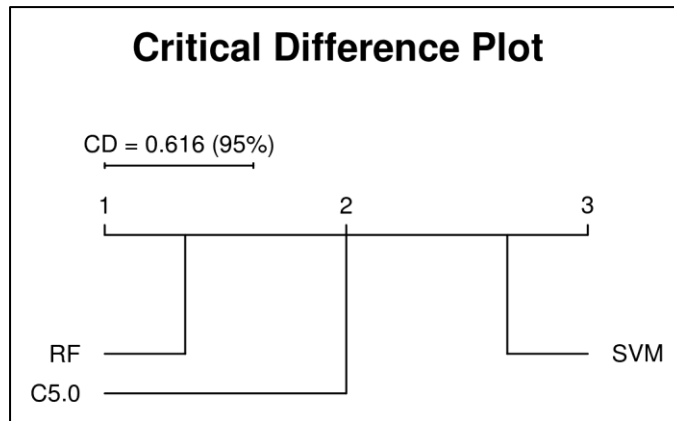


Figure 4.5: Critical differences plot of the pairwise Nemenyi comparison test for the full datasets 10-folds Cross-Validation experiment.

Table 4.7 presents the medians of each algorithm's performance (G-Mean Accuracy) as well as their mean ranks. **Figure 4.5** shows that Random Forest is highly ranked, whereas SVM scored the lowest.

	C5.0	RF	SVM
Median	0.857	0.895	0.862
1 st Quantile	0.716	0.658	0.563
3 rd Quantile	1.000	1.000	0.995
Mean Rank	2.000	1.333	2.667

Table 4.7: Algorithms' medians and mean ranks for the full datasets 10-folds Cross-Validation experiment.

4.5.2 Subset-to-Subset (File-to-File)

This experiment aimed to evaluate the capabilities of different algorithms (C5.0, Random Forest, SVM) in classifying instances with different statistical properties to those used in producing the models. This evaluation was conducted by measuring each model's performance in terms of G-Mean Accuracy.

This section shows the results of using a subset of data from each dataset (gureKDD, SEA, and AGR) in model generation using the three algorithms, where each generated model was used to evaluate the remaining parts of the dataset. For each subset, a model was generated, and the optimal cutoff and G-Mean accuracy were reported using 10-folds Cross-Validation. Also, the predicted G-Mean accuracy of each remaining subset was reported where the models' optimal

cutoff was used and compared with the G-Mean accuracy of the adapted cutoff for that subset. This experiment aimed to analyse the effect of cutoff adaptation on the evaluated data and to assess how such adaptations would compare with the use of the optimal cutoff on model performance.

Plots of the G-Mean accuracy present the performance of the prediction model (MDL_k), that was trained using $File_k$, on the files in the dataset ($File_{i \neq k}$) that were not used in producing that model. In **Figure 4.6**, **Figure 4.7** and **Figure 4.8** each model's performance, based on the Cross-Validation technique, is illustrated with a solid line; other individual performance evaluations are depicted with dotted lines.

4.5.2.1 C5.0:

Algorithm C5.0 has the worst performance on the first file in the gureKDD dataset even at Cross-Validation evaluation during the model generation stage (**Figure 4.6**). This is due to the fact that this file has the least number of attacks and is the most imbalanced of the files. It consists of only 21 attacks which formed 0.0118% of the total traffic in that file. Therefore, the generated model using this file was not able to predict any instances in other files. Where the number of attacks in other files increased with a proportionate balance, the model performances improved under this algorithm.

As the SEA and AGR datasets are composed of only six files each, there is no illustration of model 7 for these datasets in the plot. Generally, applying this approach followed the same pattern as the first experiment (10-folds Cross-Validation), where performance on gureKDD

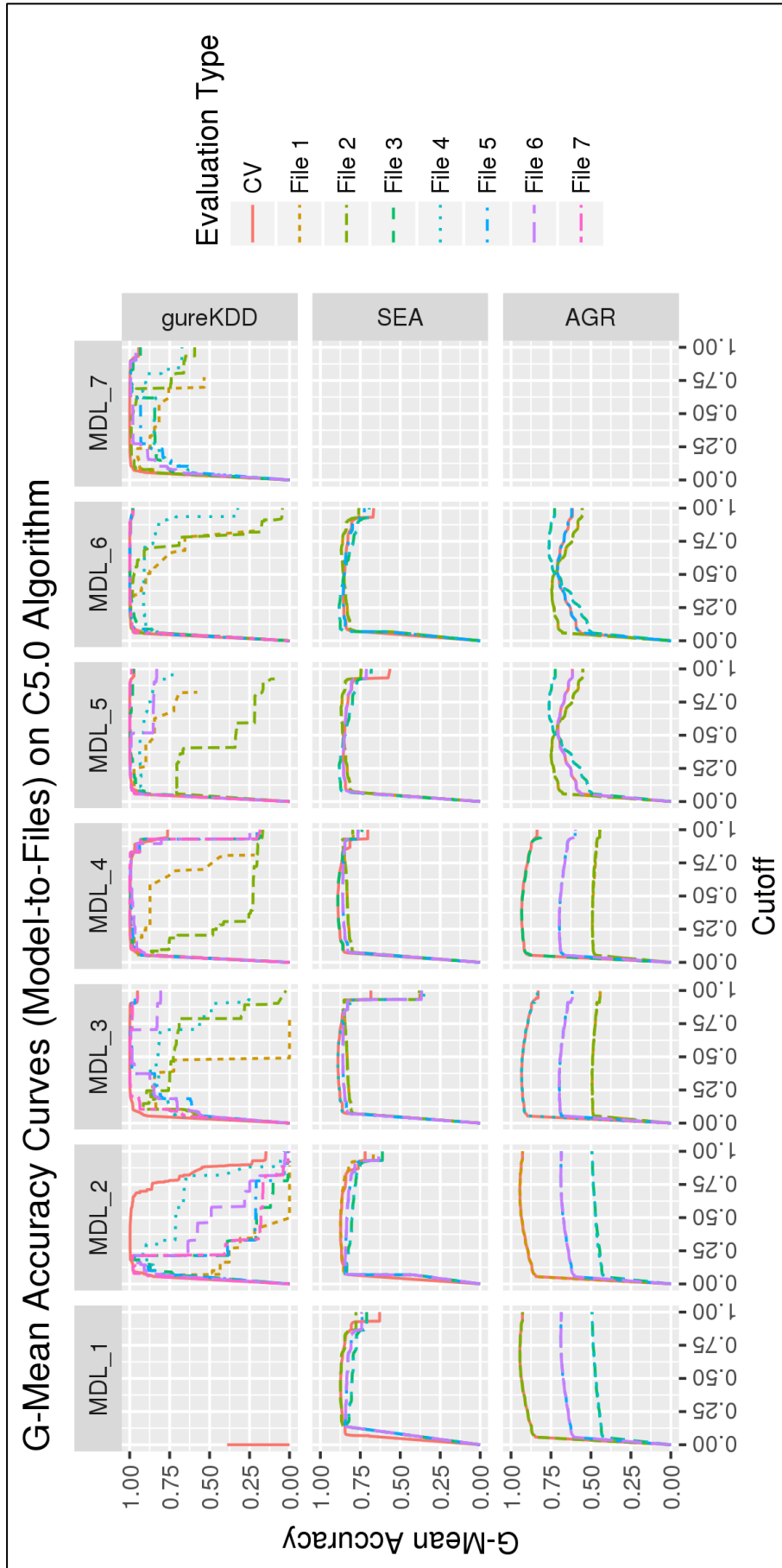


Figure 4.6: G-Mean Accuracy Curves for the C5.0 Algorithm (for gureKDD data see Table A.1, SEA data see Table A.2 and AGR data see Table A.3)

resulted in the highest accuracy followed by the SEA dataset; the worst performing dataset was the AGR.

Also in both datasets (SEA and AGR), models performed best when files exhibited the same statistical properties, denoted in these experiments by the same generating functions. For example, where MDL_1 used File 1 as training data it predicted instances in File 2 with a high performance and *vice versa* (as both files were generated using the same function). This is also applicable for Files 3 and 4 in predicting each other. Where files contain mixed behaviours, the prediction performance dropped sharply.

Section A.1 in **Appendix (A)** presents the results of each model on every file generated by each of the different algorithms. These tables show that the performance of all of these models improved when the cutoff (threshold) was adapted for the evaluation dataset, rather than using a pre-calculated one.

4.5.2.2 Random Forest (RF)

It was expected that Random Forest (RF) would perform well on the first file of the gureKDD dataset despite its low number of attack connections. Unlike C5.0, the performance of RF in modelling this file was linked to the bootstrap stage, where instances were sampled from the population with replacement. This means that duplicates of the 21 attack connections were sampled many times which increased the predictability of the built trees (**Figure 4.7**).

After careful examination of the results, as presented in **Appendix (A)** (see **Section A.2**), one can see, especially in the synthetic data (SEA and AGR), that when a testing file has similar statistical properties to the model, its performance will not increase much even after cutoff adaptation. However, when it has different statistical properties, the adaptation process boosts the prediction leading to an accurate evaluation of a model's performance.

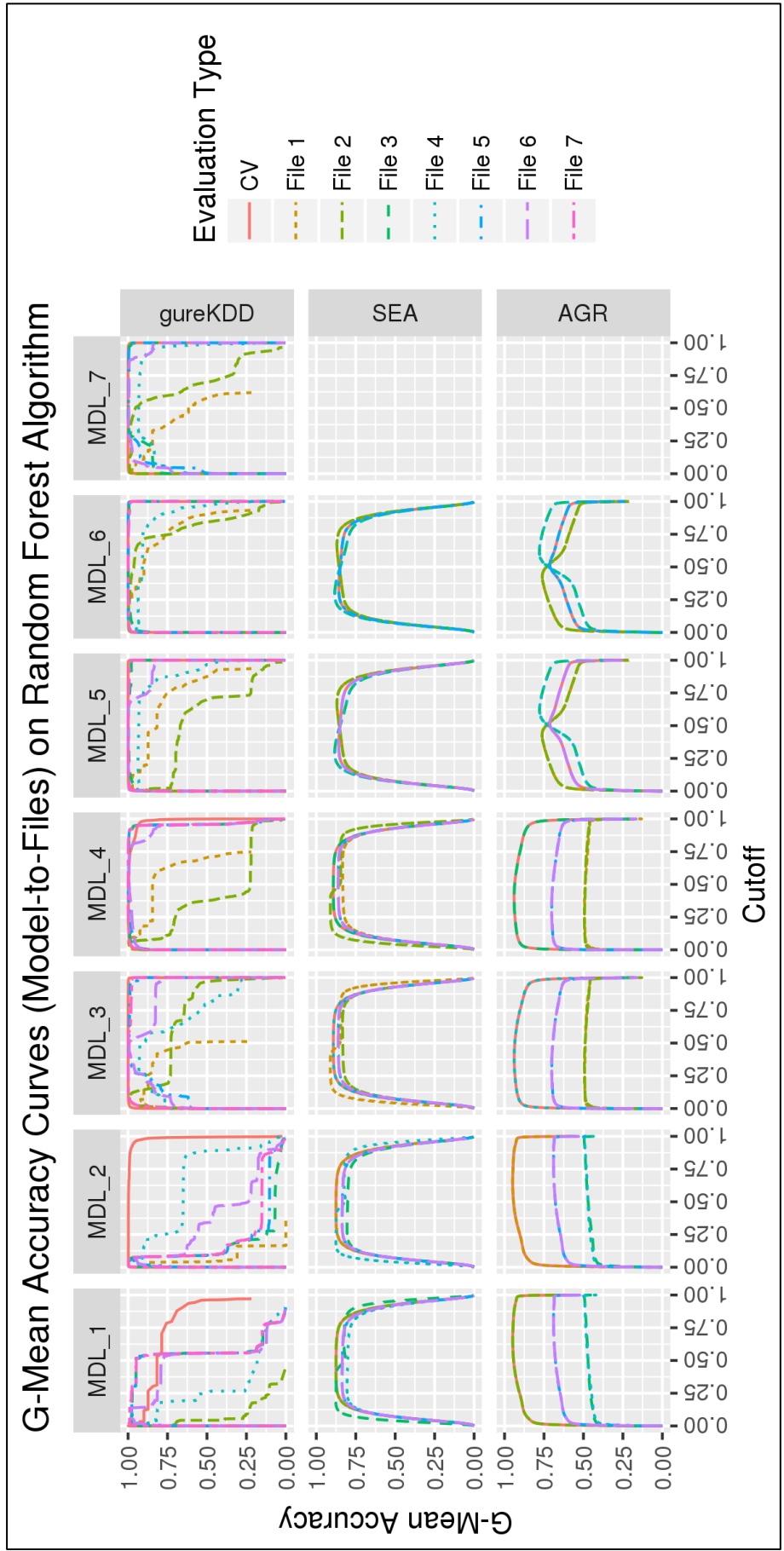


Figure 4.7: G-Mean Accuracy Curves for Random Forest Algorithm (for gureKDD data see Table A.4, SEA data see Table A.5 and AGR data see Table A.6)

Furthermore, the effect of the adaptation process was more tangible in gureKDD than in the synthetic data, as this dataset exhibited both different patterns and varying statistical properties between files. For example, **Table A.4** in **Appendix (A)** shows that MDL_1 , which was trained on File 1, reached a G-Mean accuracy of 67.33% on File 5 when the original cutoff (threshold) of the model was used, but applying the adaptation process to this threshold increased its performance to 99.37%.

4.5.2.3 SVM

SVM performed the worst on the AGR dataset in comparison to the other algorithms (**Figure 4.8**). This could have been the result of the non-linear nature of this dataset as discussed previously. The non-linearity was not picked up by the SVM linear implementation used in these experiments. In general, the cutoff (threshold) adaptation showed a similar effect in improving the models' performances compared to using the model's optimal threshold.

The findings of the experiments in this section illustrate the importance of the adapted cutoff value to the data-model pairs in achieving an accurate reading of each model's performance. To assess the significance of any differences between the models before and after the adaptation of the threshold, the Friedman test was performed.

In order to use the Friedman test, the G-Mean accuracy values were formatted into a matrix as **Table 4.8** shows the arrangement. The unshaded part of **Table 4.8** was submitted to the Friedman test function. Every row in this matrix contains the G-Mean accuracies after testing MDL_i on $File_{j \neq i}$, where two values for every algorithm were recorded before (original) and after (adaptive) the threshold adaptation. All the values in this matrix are presented in the tables of in **Appendix (A)**; each model's performance before the cutoff adaptation is denoted by *Model Threshold G-Mean Accuracy (MA)*, and after adaptation as *File Threshold G-Mean Accuracy*

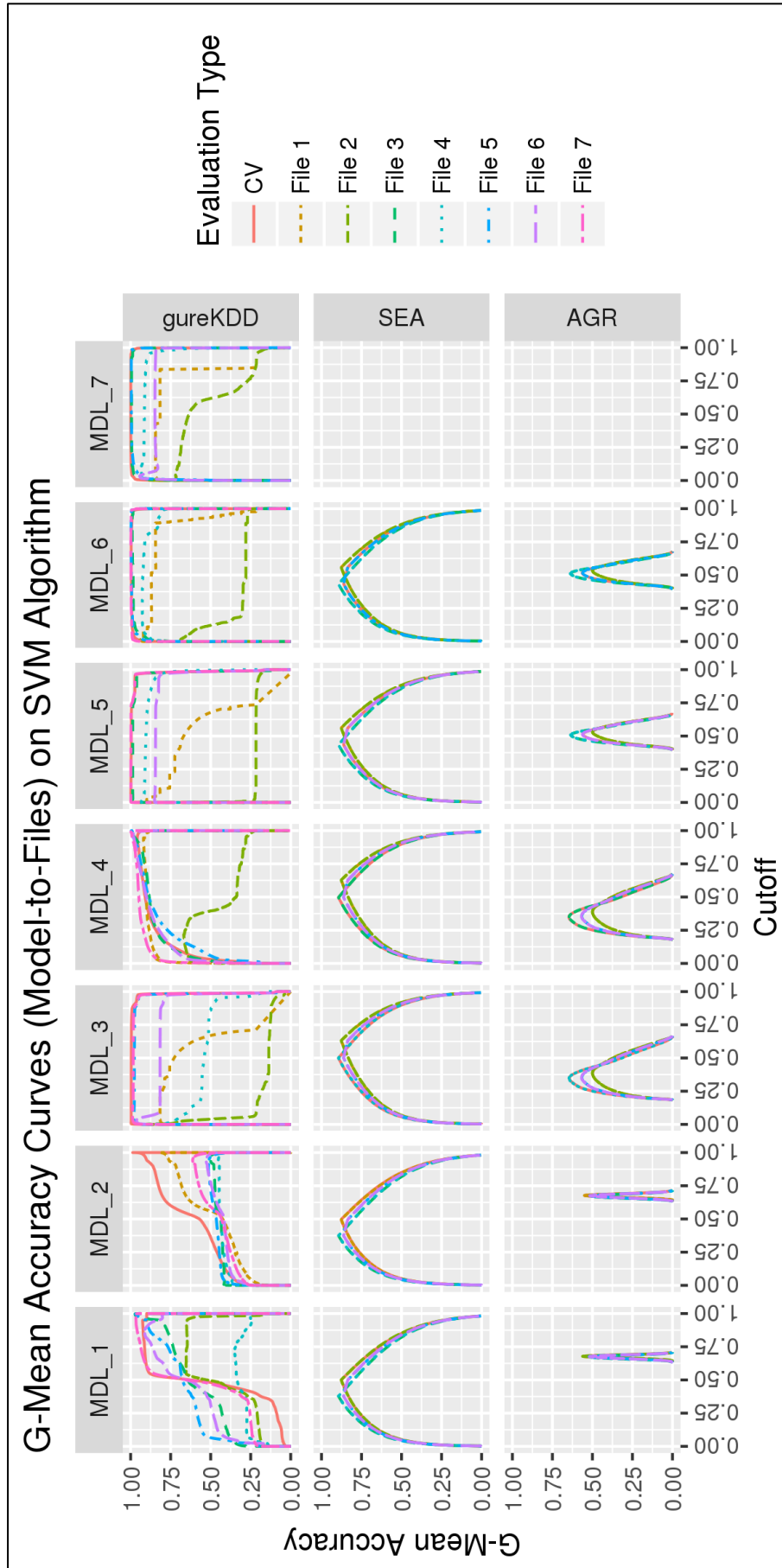


Figure 4.8: G-Mean Accuracy Curves for the SVM Algorithm (for gureKDD data see Table A.7, SEA data see Table A.8 and AGR data see Table A.9)

(FA). For every algorithm there was a total of 102 different measures for each treatment (before and after threshold adaptation). The statistical data were composed of 42 measures (7 models \times 6 evaluations) for gureKDD, and 30 measures (6 models \times 5 evaluations) for each of the SEA and AGR datasets. Hence, the resulting matrix is 102×6 .

Dataset	Model	Test file	C5.0		SVM		RF	
			Original Threshold	Adaptive Threshold	Original Threshold	Adaptive Threshold	Original Threshold	Adaptive Threshold
gureKDD	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	MDL _i	File _{j#i}	⋮	⋮	⋮	⋮	⋮	⋮
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
SEA	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	MDL _i	File _{j#i}	⋮	⋮	⋮	⋮	⋮	⋮
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
AGR	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
	MDL _i	File _{j#i}	⋮	⋮	⋮	⋮	⋮	⋮
	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Table 4.8: Models G-Mean accuracies arrangements for Friedman’s test for phase two of the experiments.

Friedman’s test was used to assess whether the difference between the different algorithms was significant before and after threshold adaptation. The tested hypothesis was, “*there is no statistically significant difference in model performances (G-Mean accuracies) before and after cutoff (threshold) adaptation between the different algorithms*”. This test revealed that there was a significant difference between the different algorithms before and after threshold adaptation, $\chi^2(5) = 217.7$, $p = 0.000 < 0.05$.

To identify which algorithms were different, a Nemenyi post-hoc test was carried out to calculate the pairwise comparisons. **Table 4.9** shows the results of this test.

		C5.0		SVM		RF
		Original Threshold	Adaptive Threshold	Original Threshold	Adaptive Threshold	Original Threshold
C5.0	Adaptive Threshold	0.000	-	-	-	-
SVM	Original Threshold	0.642	0.000	-	-	-
	Adaptive Threshold	0.000	0.999	0.000	-	-
RF	Original Threshold	0.029	0.023	0.000	0.011	-
	Adaptive Threshold	0.000	0.000	0.000	0.000	0.000

Table 4.9: Results of the pairwise Nemenyi comparison test for the cutoff (threshold) adaptation experiment.

Figure 4.9 presents the critical differences between the different algorithms before and after cutoff adaptation as a plot. The plot shows that when the cutoff was adapted for the evaluated dataset, the SVM and C5.0 algorithms were no different to each other. They showed the same behaviour even when cutoff adaptation was not performed, but the cutoff adaptation increased their performances (G-Mean Accuracy). In general, all algorithms were ranked higher when cutoff adaptation was performed, with the Random Forest algorithm always outperforming the other two.

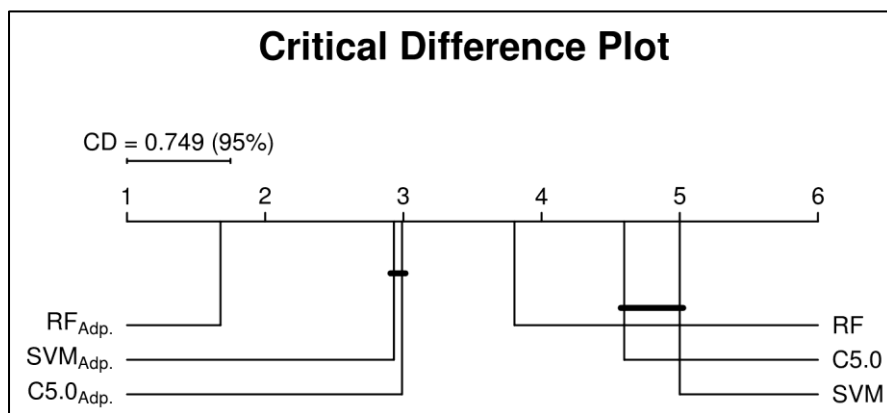


Figure 4.9: Critical differences plot of the pairwise Nemenyi comparison test for the cutoff (threshold) adaptation experiment.

Table 4.10 presents the median of every algorithm’s performance (G-Mean Accuracy) based on the threshold adaptation effect. **Figure 4.9** illustrates the mean ranks presented in **Table 4.10** as a plot.

	No Threshold Adaptation			Adaptive Threshold		
	C50	RF	SVM	C50	RF	SVM
Median	0.832	0.849	0.821	0.861	0.884	0.862
1 st Quantile	0.689	0.721	0.515	0.746	0.787	0.566
3 rd Quantile	0.877	0.939	0.888	0.949	0.991	0.903
Mean Rank	4.598	3.804	5.000	2.990	1.676	2.931

Table 4.10: Algorithms’ medians and mean ranks based on threshold adaptation effect.

4.6 Limitations

Although the experiments in this chapter have shown the advantages of adapting the cutoff (threshold) for the evaluated data, there are a number of noticeable limitations. The analysis focuses on binary classification problems with no attempt made to extend the findings to multi classification problem, i.e. type of attacks (DOS, Port Scan, SQL injection, etc.). The analysis concentrated on the batch-based classification problem because this is the focus of this thesis.

Other limitations are related to the number of the algorithms that were evaluated and the choices made with respect to the implementation of some of these algorithms. For example, the SVM linear implementation was used in these experiments, which could explain the poor performance of this algorithm on some datasets, such as AGR. This work could have been extended to use some kernels to transform the dataset into higher dimensions to improve SVM's performance. However, due to time limitations, it is planned that this task will be undertaken sometime in the future.

The algorithms were run using their default settings, with no tuning of their parameters, to improve the models' performance. For example, algorithm C5.0 was run with 10-trials only although an increased number of trials could have resulted in an improved performance. While parameter settings could be regarded as an engineering issue, future studies could be conducted to analyse the effect of different settings on the cutoff (threshold) adaptation task.

Another serious limitation of these experiments was the number of datasets used; only three datasets were used in this analysis including a very old dataset i.e. gureKDD. This limitation is addressed in the next chapter where a newer and more recent dataset is generated and tested. Despite the state and number of datasets used, an extended analysis was undertaken as a result of the multiple model evaluations performed by dataset partitioning.

4.7 Summary

This chapter has presented an analysis of the adaptive cutoff (threshold) approach, which provides an accurate reading of the performance of models generated on a training data and those used to evaluate data with different statistical properties (concept drift). The results from this analysis demonstrated the following:

- An adaptive cutoff (threshold) approach results in better classification performance than a fixed threshold.
- Using a single cutoff (threshold) will lead to misleading results, which could result in a decision to terminate a good prediction model which only required some tuning.
- This approach may not improve a model's performance when the testing data exhibits the same statistical properties as the training data.

In **Chapter 6** a more in-depth analysis is undertaken to assess the applicability of this approach to a more domain specific problem (network ID), that is, where a more recent and much more realistic dataset has been generated. In that chapter, a thorough analysis is outlined of the effect of different feature set sizes and the data balance on the proposed approach.

Chapter Five

UNB ISCX 2012 Dataset Transformation

After identifying problems with the **KDD 1999** dataset - as outlined in Al Tobi and Duncan [1] - it was necessary to find an alternative dataset. With no satisfactory dataset available, it was decided to generate a new one. This chapter sets out how a new dataset was generated by transforming the **UNB ISCX 2012 Intrusion Detection Evaluation Data Set** [14], which will be referred to throughout the thesis as **ISCX2012**. The result of this transformation was a structured dataset (named **STA2018**) in which every record described each connection (between two hosts) using **550** features [549 independent variables plus one dependent (class) variable].

The main reasons to generate this dataset are to have as clean and as validated a dataset as possible. The resultant dataset (**STA2018**) captured every ICMP, TCP and UDP packet in the ISCX2012 dataset. It also maintained the chronological order of every connection which will help future studies as the ordered data can capture different traffic behaviour and network patterns. It also provided balanced labelled connections by generating synthetic samples, which were also identifiable. In addition, each record in the resultant dataset can easily be linked back to the original ISCX2012 dataset, as the timestamp and host addresses were maintained.

5.1 ISCX2012 Dataset Description

The generation of the ISCX2012 dataset followed a systematic approach, in which datasets are created using profiles of real networks, as proposed by Shiravi *et al.* [14]. These profiles are composed of general representations of network behaviours. For instance, the profile of a host will consist of the distribution of exchanged packets, connection durations, *etc.* Such profiles are used by agents that are programmed to generate traffic, which will simulate the traffic of a real network. These profiles can be shared without privacy concerns so other researchers can then generate new traffic based on these profiles.

Two main profiles were generated to produce the network traffic, *alpha-profile* and *beta-profile*. Alpha-profiles were used to describe attack scenarios, whereas beta-profile were used to profile network events and behaviours (hosts and services).

After profiles for known applications such as, HTTP, FTP, SMTP/IMAP and SSH, had been generated, they were executed by agents within an infrastructure to generate the ISCX2012 dataset. The network used to generate the traffic was formed of six main LANs. Four different LANs accommodated 21 workstations (using different versions of Windows) which formed the interconnected network. The fifth LAN was dedicated to the servers to provide four important services: web; email; DNS and Network Address Translation (NAT) services. The sixth LAN was used for administration purposes such as, monitoring and controlling the network setup. To eliminate noise, all of the traffic in the sixth LAN was not captured. All of the transmitted traffic was mirrored to three devices that provided *redundant capturing* (e.g. tcpdump); *Intrusion Detection Systems (IDS)* (e.g. Snort); *IDS management systems* (e.g. QRadar, OSSIM); and *visualization systems* (e.g. ntop).

The traffic capturing process lasted seven days, from Friday (11/June/2010) to Thursday (17/June/2010). This simulation created seven PCAP files with labelled flow (XML) files (see **Table 5.1**). Every XML file contained the label of each connection in the corresponding PCAP

file. This simulation consisted of four main attack scenarios: inside network infiltration; HTTP Denial of Service (DoS) using the `Slowloris` tool; a Distributed Denial of Service (DDoS) attack using an IRC Botnet; and brute force SSH.

Date	PCAP files	Labelled flow (XML) files	Description ¹⁹	Size (GB)
Fri 11/Jun/2010	testbed-11jun.pcap		Normal Activity. No malicious activity	16.1
Sat 12/Jun/2010	testbed-12jun.pcap	TestbedSatJun12Flows.xml	Normal Activity. No malicious activity	4.22
Sun 13/Jun/2010	testbed-13jun.pcap	TestbedSunJun13Flows.xml	Infiltrating the network from inside + Normal Activity	3.95
Mon 14/Jun/2010	testbed-14jun.pcap	TestbedMonJun14Flows.xml	HTTP Denial of Service + Normal Activity	6.85
Tue 15/Jun/2010	testbed-15jun.pcap	TestbedTueJun15-1Flows.xml TestbedTueJun15-2Flows.xml TestbedTueJun15-3Flows.xml	Distributed Denial of Service using an IRC Botnet	23.4
Wed 16/Jun/2010	testbed-16jun.pcap	TestbedWedJun16-1Flows.xml TestbedWedJun16-2Flows.xml TestbedWedJun16-3Flows.xml	Normal Activity. No malicious activity	17.6
Thu 17/Jun/2010	testbed-17jun.pcap	TestbedThuJun17-1Flows.xml TestbedThuJun17-2Flows.xml TestbedThuJun17-3Flows.xml	Brute Force SSH + Normal Activity	12.3

Table 5.1: UNB ISCX 2012 dataset files.

Unfortunately, the ISCX2012 dataset had a limited number of attack types as it only contained these four named scenarios. Also, the traffic labels did not provide any information about the attack type as it used binary labelling {Normal and Attack} and made no further distinctions. It also offered a limited number of services in comparison to DARPA 1998. However, the ISCX2012 contained traffic that was more plausible for a modern network. This was especially the case in relation to traffic load and exchanged content as it used a realistic network set-up, and its traffic was generated using profiles from real networks as discussed earlier. Its traffic included a complete capture of all payloads as well as all of the interactions within and between LANs [14].

The ISCX2012 dataset has been made publicly available by its authors and a copy was obtained and used for the transformation outlined in this chapter. **Table 5.1** lists the files that were provided within the ISCX2012 dataset. For every traffic trace (PCAP) file there was one or more labelled flow (XML) files, apart from the Friday file (11/Jun/2010) as it consisted of

¹⁹ <http://www.unb.ca/cic/datasets/ids.html>

normal traffic only. Each XML file described every connection in the PCAP file using 19 features, where the label (*Normal* or *Attack*) was provided by the <Tag> element.

5.2 Transformation Process

There were five phases to the transformation of the ISCX2012:

1. **Basic-features extraction:** every PCAP file was processed using Bro software [16] to extract 193 features for every ICMP, TCP and UDP connection. These features consisted of information that can be extracted from frame and packet headers such as the source and destination IP addresses and ports, connection duration, transport protocol *etc.*
2. **Validation and connection labelling:** the accurate capture of every (ICMP, TCP, UDP) packet in every PCAP file was validated, then every processed connection (in the PCAP files) was matched to its corresponding flow in the XML file using the label provided {Attack, Normal}.
3. **Extend the basic-features:** every connection was processed to derive two sets of features (*time-based* and *connection-based*). Deriving these features depended on the chronological order of the original connections. Onut's feature classification schema [15] was used in this phase.
4. **Balance:** synthetic records (connections) were generated to balance the number of Normal and Attack connections in the dataset. This balancing phase used the SMOTE algorithm [207] (see **Section 5.2.4**).
5. **Clean up:** any useless features were removed, before source and destination zone features were added, to reduce the large address space.

The resultant dataset (**STA2018**) can be used as a balanced version or can be focused on the original version of the data, as every connection is uniquely distinguishable and all of the synthetic connections are identifiable.

Each one of the phases outlined above is described and discussed in more detail below.

Figure 5.1 illustrates the flow of this process, where the numbers in the output result of every phase represent the number of features in the processed resultant data.

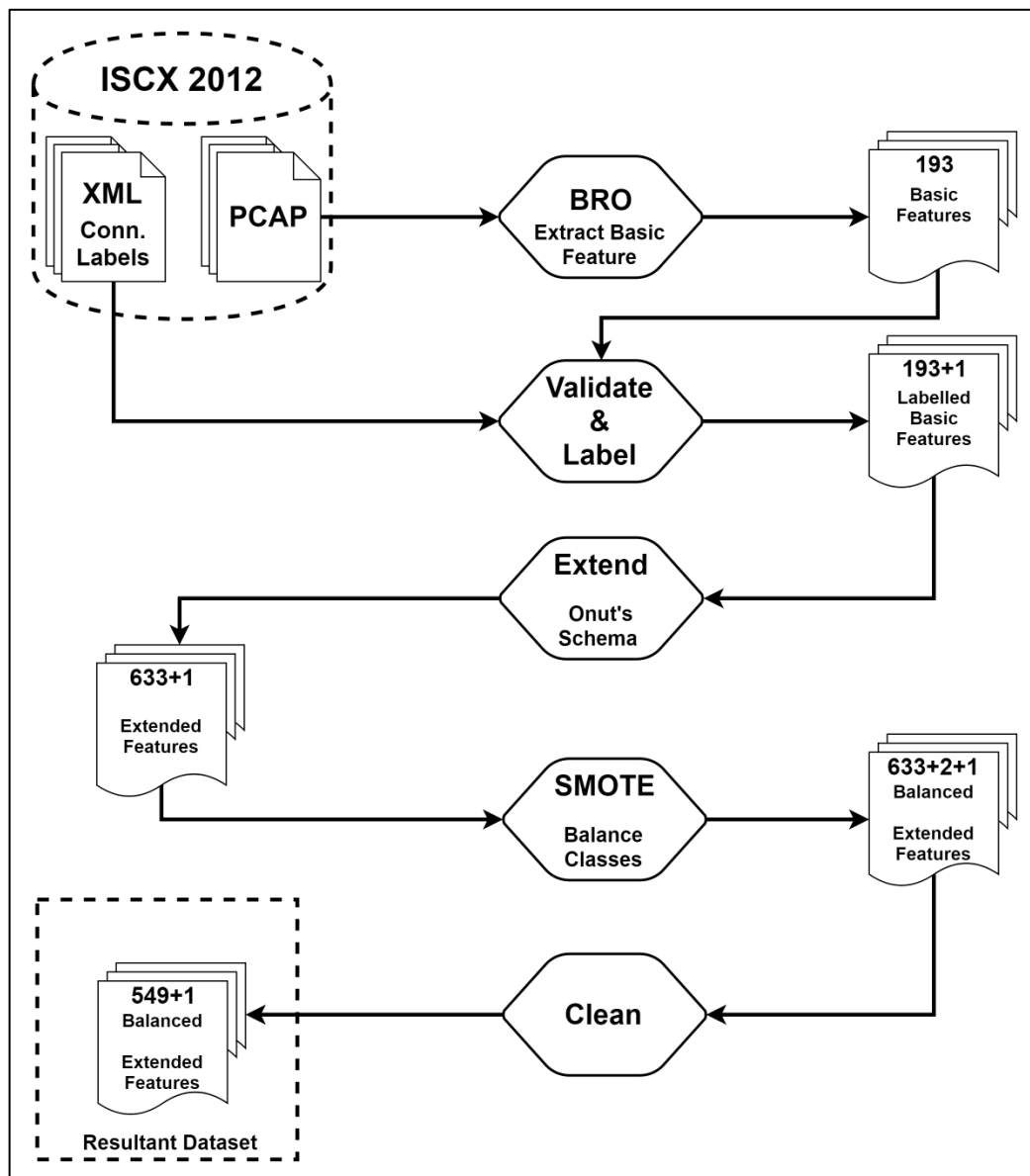


Figure 5.1: Dataset preparation phases

5.2.1 Basic features extraction

As shown in **Figure 5.1**, during the first phase every PCAP file in the ISCX2012 dataset [14] was processed in order to extract all the connection basic features for the ICMP, TCP and UDP traffic. A Bro [16] script was implemented for this process. The extraction of these features was limited to header parameters (within a connection) and did not go into payload level for several reasons, including encrypted traffic issues and privacy concerns.

At this stage, a total of 193 features were extracted [Features {1-2, 4-5, 7-12, 15-144} in **Appendix (D)**]. These features ranged from the addresses and ports numbers of the hosts involved in the connection set-up, to connection duration and the number of exchanged packets and byte sizes. Some of these features extracted the same information from different levels of the Bro IDS [16]. For example, connection duration, {**duration** and **bro_duration**}, and the number of packets and byte features were extracted using the packet and connection event handlers in Bro. As Bro uses different engines to process this information at different levels, it was necessary to include both views in the features set.

These extracted basic features were divided into the following main groups:

- IP protocol type:
 - IPv4 connections' related features (32 features, three of which were deleted – see **Section 5.2.5**);
 - IPv6 related features (34 features, 27 of which were deleted – see **Section 5.2.5**).
- Transport protocol type:
 - ICMP related features (2 features, one of which was deleted – see **Section 5.2.5**);
 - TCP related features (62 features, 21 of which were deleted – see **Section 5.2.5**);
 - UDP related features (8 features).

Where a connection was an IPv4 type, then all IPv6 related features were set to zero and *vice versa*. This same process was applied to the transport protocol type features. For example, where a connection was a TCP type, then all ICMP and UDP features were set to zero.

In this phase, Bro's default setting for a TCP connection's start time had been overridden to use the timestamp of the first SYN packet. Bro's default settings use the last SYN packet in the hand-shake phase to mark the start time of connections. All other settings were set to the default, including any TCP connection timing out after being idle for more than 5 minutes.

In the design of the Bro script, TCP connections are treated as **statefull** connections with the Bro's default engine settings deciding which sequence of packets are grouped as a single connection. Although UDP is not a session-based protocol, Basu *et al.* (2001) [345] suggested the construction of a session view for such traffic, as it would be useful to detect low-profile probes and novel DoS attacks. Therefore, the Bro UDP connection event handler was used which bundles together a sequence of exchanged UDP packets between two hosts into one connection, based on some heuristic and service profiles. All ICMP traffic was treated as **stateless** connections and every packet was treated as a single connection. Although Bro is able to aggregate a sequence of exchanged ICMP packets into a single connection if they meet certain criteria, it was decided to use the stateless nature of the traffic in the generated dataset. This process is similar to the KDD 1999 design in profiling the DARPA 1998 network traffic.

5.2.2 Validation and labelling

This stage aimed to validate the information extracted by Bro scripts. As such, a Perl script was implemented to count all existing packets and their types in the PCAP files and to compare these totals with what had been processed by Bro (see columns Bro and PCAP in **Table 5.2**). This comparison showed that the transformed data captured every ICMP, TCP and UDP packet correctly, as the totals matched.

			BRO	PCAP	XML		
11-Jun	IPv4	ARP	-	14501	-	-	IPv4
		ICMP	4,316	4,316	-	icmp_ip	
		IGMP	-	14	-	igmp	
		TCP	21,361,138	21,361,138	-	tcp_ip	
		UDP	306,793	306,833 (40 Frag.)	-	udp_ip	
	IPv6	HOPOPT	-	12	-	ip	IPv6
ICMPv6		12	0	-	ipv6icmp		
UDP		1,459	1,459	-	0.0.0.0		
12-Jun	IPv4	ARP	-	13,088	-	-	IPv4
		ICMP	343	343	343	icmp_ip	
		IGMP	-	60	60	igmp	
		TCP	5,816,735	5,816,735	5,796,554	tcp_ip	
		UDP	142,158	142,230 (72 Frag.)	143,105	udp_ip	
	IPv6	HOPOPT	-	58	58	ip	IPv6
		ICMPv6	60	2	2	ipv6icmp	
		UDP	1,464	1,464	1,457	0.0.0.0	
		HOPOPT	-	39	78	ip	
13-Jun	IPv4	ARP	-	14,771	-	-	IPv4
		ICMP	2,299	2,299	4,598	icmp_ip	
		IGMP	-	43	86	igmp	
		TCP	5,613,966	5,613,966	11,169,824	tcp_ip	
		UDP	130,818	130,822 (4 Frag.)	263,702	udp_ip	
	IPv6	HOPOPT	-	39	78	ip	IPv6
		ICMPv6	39	0	0	ipv6icmp	
		UDP	1,209	1,209	2,412	0.0.0.0	
		HOPOPT	-	10	10	ip	
14-Jun	IPv4	ARP	-	14,339	-	-	IPv4
		ICMP	2,130	2,130	2,126	icmp_ip	
		IGMP	-	14	14	igmp	
		TCP	8,924,769	8,924,769	8,922,334	tcp_ip	
		UDP	705,942	705,965 (23 Frag.)	698,797	udp_ip	
	IPv6	HOPOPT	-	10	10	ip	IPv6
		ICMPv6	10	0	0	ipv6icmp	
		UDP	1,426	1,426	1,429	0.0.0.0	
		HOPOPT	-	78	78	ip	
15-Jun	IPv4	ARP	-	14,502	-	-	IPv4
		ICMP	14,031	14,031	14,035	icmp_ip	
		IGMP	-	79	79	igmp	
		TCP	33,734,618	33,734,618	33,758,426	tcp_ip	
		UDP	1,218,233	1,218,287 (54 Frag.)	1,228,442	udp_ip	
	IPv6	HOPOPT	-	78	78	ip	IPv6
		ICMPv6	78	0	0	ipv6icmp	
		UDP	1,447	1,447	1,447	0.0.0.0	
		HOPOPT	-	10	10	ip	
16-Jun	IPv4	ARP	-	14,785	-	-	IPv4
		ICMP	1,379	1,379	1,379	icmp_ip	
		IGMP	-	14	14	igmp	
		TCP	24,230,487	24,230,487	24,192,695	tcp_ip	
		UDP	345,959	345,961 (2 Frag.)	347,344	udp_ip	
	IPv6	HOPOPT	-	10	10	ip	IPv6
		ICMPv6	10	0	0	ipv6icmp	
		UDP	1,428	1,428	1,428	0.0.0.0	
		HOPOPT	-	24	24	ip	
17-Jun	IPv4	ARP	-	16,422	-	-	IPv4
		ICMP	2,080	2,080	2,080	icmp_ip	
		IGMP	-	24	24	igmp	
		TCP	17,029,774	17,029,774	17,066,126	tcp_ip	
		UDP	260,896	260,935 (39 Frag.)	262,237	udp_ip	
	IPv6	HOPOPT	-	24	24	ip	IPv6
		ICMPv6	24	0	0	ipv6icmp	
		UDP	1,425	1,425	1,418	0.0.0.0	
		HOPOPT	-	24	24	ip	

Table 5.2: Packet counts comparison for all Bro, PCAP and XML files

This stage also aimed to match every processed connection with its original label from the XML files of the ISCX2012 dataset. This was not a straight forward task as a number of problems were identified with the XML files during the validation stage. These challenges were addressed on a case by case basis. Following this stage, there were **193** features and **one** label (class) feature. A detailed list of the validation steps and each of the problems that arose is presented in **Section 5.3**, which also discusses how these problems were addressed.

5.2.3 Extending the features space

During this phase, a feature engineering technique was applied, to derive new features, based on the extracted basic features of various types of connections within a network. These newly derived features aimed to find similarities between different connections. Therefore, this phase has used Onut's feature classification schema [15], which provides large amounts of information by deriving features, ranging from packet-level to connection-level views of network traffic. The part of this schema related to derived connection-level features (the shaded parts in **Figure 5.2**) was adopted in this work in order to extend the basic features by deriving **Time-based** and **Connection-based** features.

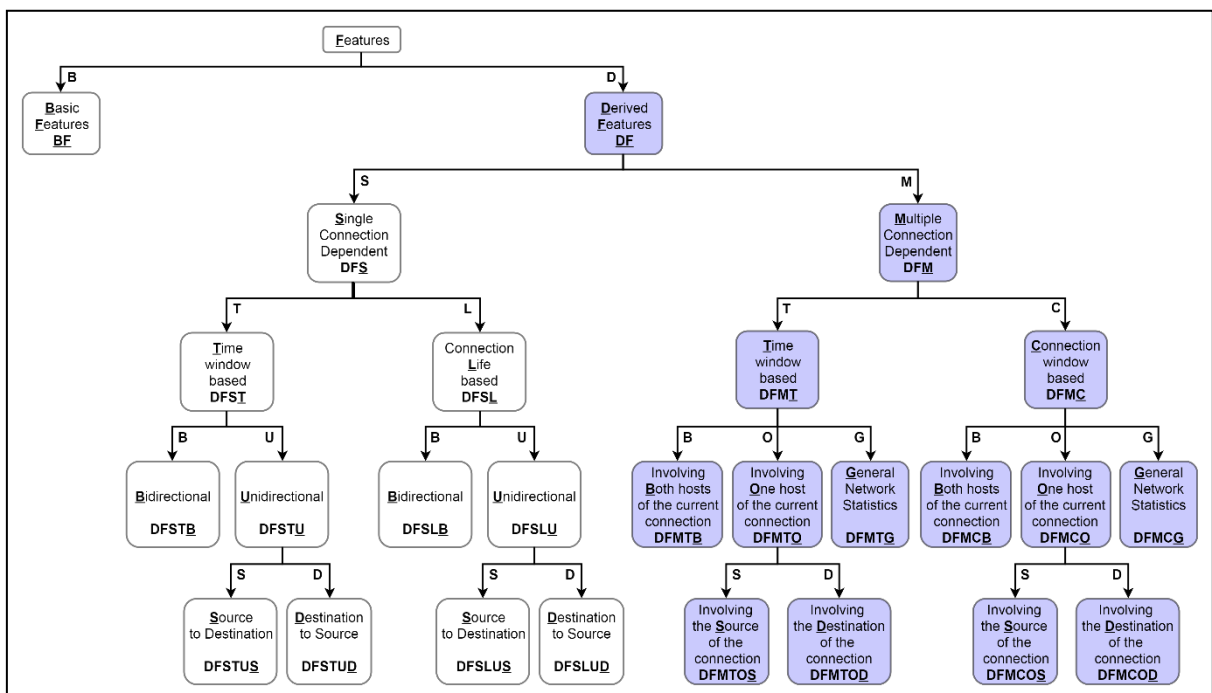


Figure 5.2: Onut's Feature Classification Schema [15]

Time-based features aimed to detect bursty attacks and used a sliding window of 5 seconds to compute these derived features (220 features). Connection-based features targeted stealthy attacks and used a sliding window of 100 connections to derive the same features (220 features). This classification schema was implemented in Java as the original implementation could not be obtained from the author. This implementation used the basic features listed in **Table 5.3** to derive all 440 features. Details of every feature can be found in the feature descriptions table in **Appendix (D)**. It is worth noting that the basic features, `tcp_src_flags_URG_flags` and `tcp_dst_flags_URG_flags`, were deleted at the cleaning phase, after the extending phase, after being identified as useless.

No.	Feature name	No.	Feature name
1	<code>start_time</code>	99d	<code>tcp_src_flags_URG_flags</code>
2	<code>src_ip</code>	100	<code>tcp_src_flags_ACK_flags</code>
4	<code>src_prt</code>	101	<code>tcp_src_flags_PSH_flags</code>
5	<code>dst_ip</code>	102	<code>tcp_src_flags_RST_flags</code>
7	<code>dst_prt</code>	103	<code>tcp_src_flags_SYN_flags</code>
10	<code>protocol</code>	104	<code>tcp_src_flags_FIN_flags</code>
17	<code>src_bytes</code>	120b	<code>tcp_dst_flags_URG_flags</code>
18	<code>dst_bytes</code>	121	<code>tcp_dst_flags_ACK_flags</code>
19	<code>src_pkts</code>	122	<code>tcp_dst_flags_PSH_flags</code>
20	<code>dst_pkts</code>	123	<code>tcp_dst_flags_RST_flags</code>
24	<code>bro_service</code>	124	<code>tcp_dst_flags_SYN_flags</code>
		125	<code>tcp_dst_flags_FIN_flags</code>

Table 5.3: Basic features used in Onut's schema to extend the features set

5.2.4 Balancing the dataset

As the developers of ISCX2012 attempted to produce a dataset that resembled reality, the dataset following this transformation was highly imbalanced, with far fewer **attack** connections than **normal** ones, as illustrated in **Figure 5.3**. The ratio of Normal to Attack cases had an imbalance as high as (**54,547:1**) on Day 6 (16/Jun/2010).

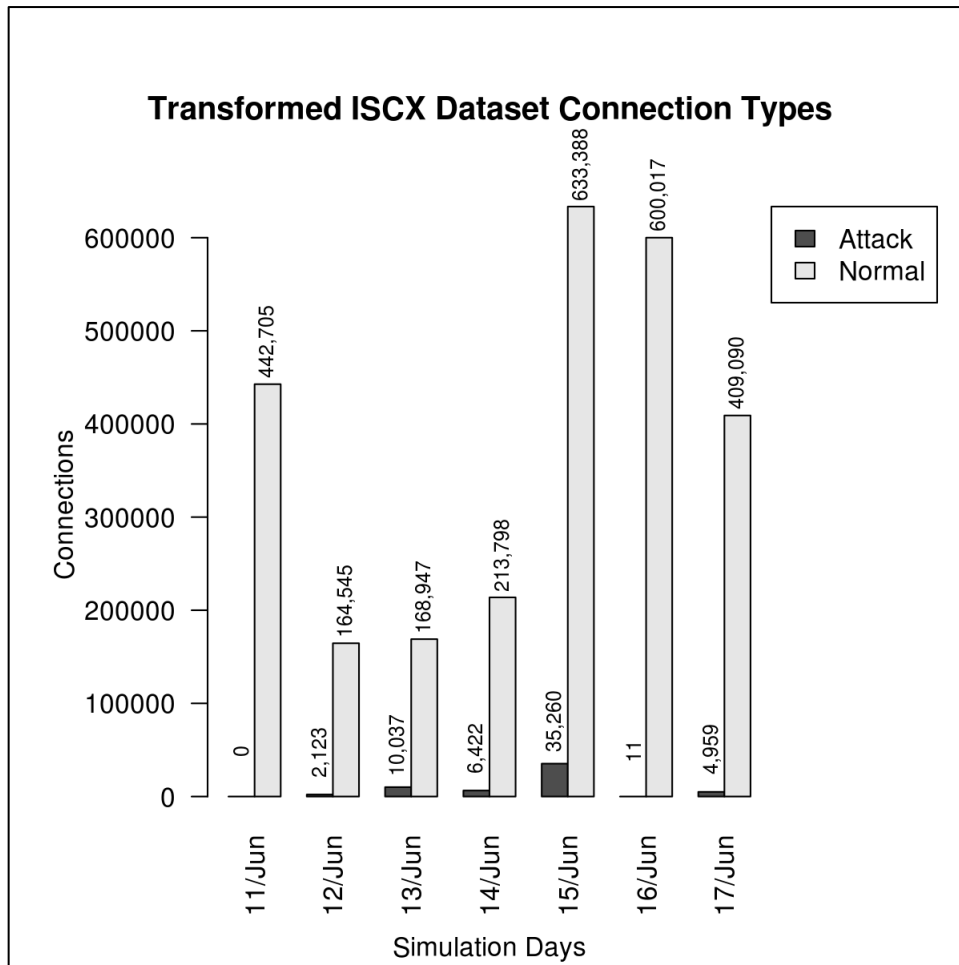


Figure 5.3: ISCX2012 number of class connections for each simulation day

Using such a dataset with this kind of imbalance could result in biased models. There are many techniques to address this issue as discussed in **Chapter 2**, including: using evaluation measures other than accuracy (such as, balanced-accuracy or G-Mean Accuracy); over or under sampling techniques; different cost (penalising) approaches; and/or the generation of new synthetic samples.

This work adopted the approach of generating synthetic samples, as it provided the flexibility for researchers to include or omit the generated samples in future studies.

Synthetic samples were generated using the DMwR package [346, 347], which includes the Synthetic Minority Over-sampling Technique (SMOTE) algorithm [207], in R software [321]. The SMOTE algorithm works, as Chawla *et al.* [207] detailed, by randomly sampling a new

instance from the line segments that connects one of the minority (Attack) observations to any or all of its K minority nearest neighbours. To generate a new synthetic instance, as **Eq.(5.1)** shows, the algorithm calculates the difference between the minority sample and its nearest neighbour using their feature vectors. A random number (between 0 and 1) is multiplied by that difference, and the total is added to the feature vector of the minority observation under consideration.

$$x_s = r(x - x_{k_i}) + x \quad \text{Eq.(5.1)}$$

where x_s is a new synthetic instance, x is the actual minority sample, x_{k_i} is one of the K minority nearest neighbours of x and r is a random number between zero and one. This process is repeated until the targeted number of synthetic samples are generated. **Figure 5.4** illustrates an example of the SMOTE algorithm in a two-dimensional space, with the blue points representing the synthetic samples that were generated at random from the line connecting two minority (+) instances. This algorithm was executed with its default parameters, with $K=5$ nearest neighbours used to generate the new samples.

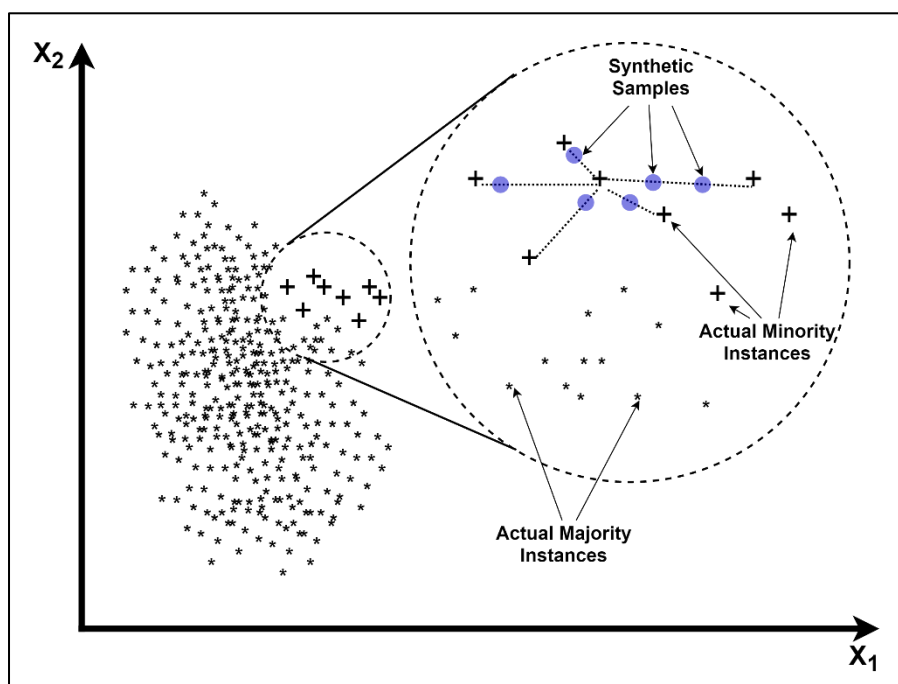


Figure 5.4: Generation of synthetic instances using the SMOTE algorithm

In this phase, the SMOTE algorithm was used to generate synthetic samples of Attack connections only. The number of these samples was the difference between the number of Normal instances and the number of Attack instances ($N_{normal} - N_{attack}$) in every simulation day except the first day (which only contained normal traffic). Any synthetic instances that duplicated an original instance were removed.

During this phase, two extra features (“**synthetic**” and “**origOrder**”) were added to the new dataset. The “**synthetic**” feature was used to identify every synthetic connection by setting its value to one (yes) or zero (no) otherwise. The “**origOrder**” feature contained the order number of every actual record (connection) as it appeared in the original data file and zero for every synthetic sample. These features were added to help researchers identify any synthetic instances but need to be removed when any learning or prediction processes are taking place.

5.2.5 *Cleaning the dataset*

The main objective of this phase was to reduce the feature space by eliminating any useless features. A quick analysis was conducted of every feature in the dataset to identify any non-changing (single value) features. During this analysis the number of unique values for every feature in each simulation file was counted. If a feature was found to contain the same value in all of the data files, or to be useful in only one data file, it was eliminated. This analysis revealed that there were 88 useless features, all of which were removed from the final dataset resulting in a total of 548 features (547 features + 1 class). **Table 5.4** lists these eliminated features and their index numbers [which also appear in the feature description tables in **Appendix (D)**].

During the last stage of this phase two more features were generated (**src_zone** and **dst_zone**), which are source and destination topological zones {“GLOBAL”, “MULTICAST”, “UNICAST”, “UNKNOWN”, “LOCAL”, “LAN1”, “LAN2”, “LAN3”, “LAN4”, “LAN5”, “LAN6”}. These two features reduced the address space of the source and destination columns, and guarded against the

	Total feature	Feature's indexes
Basic features	53	20a, 50a, 53a, 64a, 79a, 79b, 79c, 79d, 79e, 79f, 79g, 79h, 79i, 82a, 86a, 86b, 86c, 86d, 86e, 86f, 86g, 86h, 86i, 86j, 86k, 86l, 86m, 86n, 86o, 86p, 86q, 87a, 99a, 99b, 99c, 99d, 104a, 107a, 107b, 107c, 107d, 107e, 107f, 119a, 120a, 120b, 125a, 128a, 128b, 128c, 128d, 128e, 128f
Connection based features	19	372a, 375a, 376a, 395a, 400a, 405a, 405b, 405c, 405d, 450a, 450b, 455a, 505a, 505b, 510a, 533a, 539a, 539b, 539c
Time based features	16	194a, 199a, 204a, 204b, 204c, 204d, 249a, 249b, 254a, 304a, 304b, 309a, 332a, 338a, 338b, 338c

Table 5.4: Indexes of eliminated features

profiling of network traffic being too specific (biased) in relation to certain connections. The end result of this phase was a feature set of 550 features (549 features + 1 class). A complete list of the extracted and derived features, along with their descriptions, is provided in **Appendix (D)**; the deleted features have been distinguished by concatenating a sequence of alphabetic characters by their order number as well as being marked in red and italicized. (The deleted features were kept in the list of features for reference purposes.)

5.3 Details of Validation and Labelling Phase

This section details the validation and labelling phase. As the processing steps for validation and labelling were very similar, they were undertaken in one phase.

This phase validated the results from Bro to ensure every connection in the PCAP files had been extracted correctly. Every connection was labelled with the correct tag provided by the flow description files (XML).

The steps, findings, and resolutions of the validation stage are discussed next. The labelling process was conducted by mapping the connections from Bro with their matches in the XML files. This mapping exercise faced a number of difficulties that were identified during validation.

Before any validation process could take place, all of the PCAP files were pre-processed and some basic information was formatted into a tabular form. This pre-processing generated a text file with all packet directions and their counts for every PCAP file. These text files were used in the later stages to conduct the analyses discussed below. **Table 5.5** provides an example of such a text file.

<u>ip_version</u>	<u>protocol</u>	<u>source_ip</u>	<u>source_port</u>	<u>destination_ip</u>	<u>destination_port</u>	<u>packets_count</u>
4	tcp	12.180.55.140	80	192.168.2.106	2460	169
4	tcp	192.168.2.106	2460	12.180.55.140	80	109
4	tcp	12.180.55.140	80	192.168.2.106	2461	111
4	tcp	192.168.2.106	2461	12.180.55.140	80	70
4	udp	116.197.169.12	53	192.168.5.122	19195	1
4	udp	192.168.5.122	19195	116.197.169.12	53	1
4	udp	116.197.169.12	53	192.168.5.122	4807	1
4	udp	192.168.5.122	4807	116.197.169.12	53	1
4	igmp	192.168.4.120		224.0.0.22		2
4	igmp	192.168.4.121		224.0.0.22		12
4	icmp	192.168.5.122		98.124.196.1		3
4	icmp	209.210.145.86		192.168.5.122		1

Table 5.5: Example of total packets in every communication direction

It is worth noting that this example lists the packet directions not the connections between hosts. For example, in **Table 5.5**, the packet directions and counts of the first record between host (12.180.55.140:80) and (192.168.2.106:2460) could form one or multiple connections. It states that a total of 169 packets were transmitted from 12.180.55.140:80 to 192.168.2.106:2460. This formatting is useful to determine, later in the validation stage, which direction or connection group has inconsistent counts.

All XML files were then converted into a tabular format and saved in text files. This step reduced the size of files and eliminated any unwanted information ahead of the analysis. Each XML flow record was represented by the 14 features listed in **Figure 5.5**. The “duration” feature (**stopDateTime - startDateTime**) was computed using the converting script as it was not present in the XML files.

All references to the PCAP and XML files in the discussion below refer to these files.

startDateTime	:	Connection start time
stopDateTime	:	Connection finish time
Duration	:	Connection duration in seconds (computed by the script)
protocolName	:	Protocol name used by this connection {igmp, ip, udp_ip, icmp_ip, tcp_ip, ipv6icmp}
src_ip	:	IP address of the machine that started this connection (source)
src_port	:	Source port number
dst_ip	:	IP address of the destination machine
dst_port	:	Destination port number
appName	:	Application name used to establish this connection
src_pckts	:	Number of packets sent by the source machine
src_bytes	:	Total bytes sent by the source machine
dst_pckts	:	Number of packets sent by the destination machine
dst_bytes	:	Total bytes sent by the destination machine
conn_tag	:	Connection status {Normal, Attack}

Figure 5.5: Features in the flow files (XML) in all analyses.

5.3.1 Validation

In order to validate the results obtained from Bro, a two phased validation was undertaken. In the **first phase**, a general packet count was made of every IP protocol {IPv4, IPv6} versus every transport protocol {ICMP, TCP, *etc.*} for each PCAP file. A Perl script was implemented and executed to parse the PCAP files; self validation was undertaken by counting the size of every frame processed and comparing the total size of all frames with the total size of the PCAP file in order to check that every packet had been successfully processed. The results from this analysis were compared with the results from Bro (**src_packets** and **dst_packets** features) to ensure every targeted packet had been correctly captured and processed. **Table 5.2** sets out the number of packets for Bro and PCAP. It shows that Bro successfully detected all targeted packets. Minor differences, especially in the UDP connections, were the result of fragmented packets, which Bro usually reassembled before passing them on to the event handler.

Another difference arose in the ICMP traffic within the IPv6 connections. A careful analysis revealed that all of the **HOPOPT** packets (IPv6 packet with **hop-by-hop option** is being set) in the IPv6 traffic were basically ICMP packets, which should have been handled by each of the hosts receiving these HOPOPT packets. Bro's parser was therefore able to extract these

ICMPv6 packets, a fact which can be confirmed by comparing the number of packets in both protocols in **Table 5.2**.

The **second phase** aimed to provide further validation of the results obtained by Bro. In this phase, the number of packets in every direction was computed. A Perl script was implemented and executed to construct a list of all the communication directions and the number of packets transmitted in each direction using the results from Bro. They were then compared with the communication directions of the PCAP files that had been pre-processed, as the example shown in **Table 5.5**. This analysis revealed that Bro failed to determine the right direction of some UDP packets for only **two** of the UDP connections. A manual analysis of this issue revealed a bug in Bro's UDP analyser which did not handle certain UDP connections in the correct way.

5.3.2 Labelling

After validating the results from Bro, the same validation process was undertaken on the labelled flow (XML) files within the ISCX2012 dataset. This validation process revealed a number of problems with these files which are listed in **Section 5.3.3**.

These problems made the labelling process more challenging, as linking connections between PCAP and XML was not straightforward. For example, it was not possible to use time as part of the mapping keys as the XML files had used a human readable time format; this format did not extend timestamp precision - after conversion - as the ones in the PCAP files. Also the time difference between converted times and the PCAP's timestamps were inconsistent between one connection and another. Therefore, matching keys, such as the key (`start_time,src_ip:port,dst_ip:port`) was not helpful.

For these reasons and the problems discussed in **Section 5.3.3**, the strategy outlined below was developed to enable the correct mapping of connections, and hence, an accurate labelling process. This strategy assumed that every connection in the dataset (PCAP) was normal except

for those which were labelled as attacks in the XML files. The mapping process therefore focused on matching connections between PCAP and XML by their chronological order.

In this mapping process, all connections between two hosts were aggregated by a single key. The best matching key for this process was a combination of transport protocol {ICMP, TCP, UDP} and the sorted hosts' addresses and ports. As a result, every key was of the form `KEY(protocol, IP1:PORT1, IP2:PORT2)`. For ICMP traffic, only transport protocol and IP addresses were used to form the key (`KEY(protocol, IP1, IP2)`).

Listing 5.1 illustrates the labelling process for PCAP connections. Every connection in the PCAP was added to one of the connection sets mapped by one key, as presented by *lines 1-10*. The same mapping process was applied to all connections in the XML file (*lines 12-22*) to produce a key-connections map for the XML connections. Finally, as shown in *lines 24-29*, key-connections maps for all PCAP and XML connections were compared, and every connection with the key `keyi` in PCAP was mapped to its chronologically matched connection with the matching key `keyj` in XML. This mapping process is further illustrated in **Figure 5.6**.

5.3.3 Problems with labelled flow (XML) files

This section sets out the main problems that were identified with the ISCX2012 dataset. It is worth noting that all these issues were communicated to the authors, but no response was received from them. All of the problems listed below were found in the labelled flow (XML) files, which had been used to label connections in the PCAP files. Due to the diverse range of issues, most had to be addressed individually.

Problem 1: Wrong total byte value

It is not clear how the authors of the dataset calculated the “`src_bytes`” and “`dst_bytes`” (`<totalSourceBytes>` and `<totalDestinationBytes>`) for each connection in the XML files. It

Algorithm: Label PCAP connections**Input:** pcap.connections, xml.connections**Result:** labelled PCAP connections

```

1  pcap.keyMaps <- {}
2  for (conn in pcap.connections) do
3    key <- [connproto, sort(connsrc_IP, conndst_ip)]
4    if(connproto == "tcp" || connproto == "udp")
5      key <- [connproto, sort(connsrc_IP:connsrc_port, conndst_IP:conndst_port)]
6
7    id <- connid
8
9    pcap.keyMaps[key] <- pcap.keyMaps[key] ∪ {id}
10 done
11
12 xml.keyMaps <- {}
13 for (conn in xml.connections) do
14   key <- [connproto, sort(connsrc_IP, conndst_ip)]
15   if(connproto == "tcp" || connproto == "udp")
16     key <- [connproto, sort(connsrc_IP:connsrc_port, conndst_IP:conndst_port)]
17
18   id <- connid
19   label <- connlabel
20
21   xml.keyMaps[key] <- xml.keyMaps[key] ∪ {id,label}
22 done
23
24 for (keyi in pcap.keyMaps) do
25   if(keyi exists in xml.keyMaps && labels in xml.keyMaps[keyi] has Attack)
26     label chronologically matched connections between xml.keyMaps[keyi] and pcap.keyMaps[keyi]
27   else
28     label connections in pcap.keyMaps[keyi] as Normal
29 done

```

Listing 5.1: Pseudo code of connection labelling through mapping connections between PCAP and XML

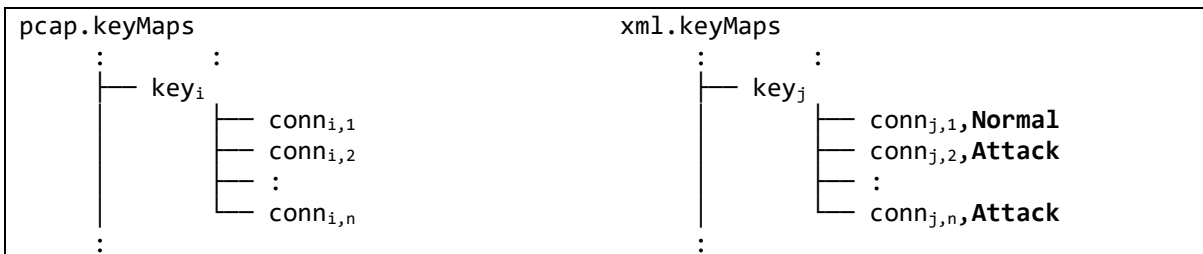


Figure 5.6: Connection matching by mapping keys

was not possible to match the values of those features for every connection with any number of bytes in the PCAP files on any level (frame bytes, IP bytes, or payload bytes). As it was not possible to use these two features in the process that matched the PCAP connections and the XML flows, they were discarded and removed from any further analysis or matching process.

Problem 2: Wrong IPv6 addresses (0.0.0.0)

During the transformation process, it was possible to identify connections with an invalid source and destination IP address as their addresses were “0.0.0.0”. For example, filtering all connections in the XML files, where the protocol name was either “ip” or “ipv6icmp” resulted in the following source and destination IP addresses:

- <destination>0.0.0.0</destination>
- <source>0.0.0.0</source>

Based on this finding, which was supported by the ICMPv6 connections in the **TestbedSatJun12Flows.xml**, it was assumed that every connection with an “0.0.0.0” IP address was an IPv6 connection. Using this assumption, matching the number of packets between PCAP and XML files became much more consistent, especially for UDP connections.

This analysis, as set out in **Table 5.2** (see columns PCAP and XML), revealed that all flows in XML with the protocol name “ip” were IPv6 flows where the packet protocol number was set to zero (HOPOPT). It also revealed that none of the IPv6 addresses (i.e. those that were present in the PCAP files) had ever appeared in any of the labelled flow files. **Table 5.6** lists all of the unique IPv6 addresses as detected in the PCAP files.

Simulation day	IPv6 Addresses set
12/Jun/2010	fe80:0:0:0:5c29:ff4:d3e7:b80d ff02:0:0:0:0:0:0:16 ff02:0:0:0:0:0:0:2 ff02:0:0:0:0:0:1:2 ff02:0:0:0:0:0:1:3
13/Jun/2010 - 17/Jun/2010	fe80:0:0:0:5c29:ff4:d3e7:b80d ff02:0:0:0:0:0:0:16 ff02:0:0:0:0:0:1:2 ff02:0:0:0:0:0:1:3

Table 5.6: IPv6 address as in PCAP files

Based on these findings all flows with the protocol name “ip” were treated as IPv6 in the matching process. It was fortunate there were not any attack connections within this group of traffic, as it would have been impossible to label them.

Problem 3: Duplicate connections in labelled flow (XML) files

Table 5.2 (see columns PCAP and XML) compares the total number of packets in each XML file with the number in each PCAP file for each day. The table highlights where there were more packets in the XML files than actually existed (italic, underlined and in red). Further investigations were then undertaken to identify any duplicate connections in those labelled flows (XML) files.

Table 5.7 shows the number of connections and any copies (duplicates) in those XML files. For example, for the **TestbedMonJun14Flows.xml** file (14/Jun/2010), the information in the table can be interpreted as follows:

- 171,322 unique connections,
- There were 26 different connections with 2 copies (instances) each,
- There were 2 other connections with 3 copies (instances) each,

In this analysis, a full connection record -with all its fields' values as in these XML files- was used to match other records. If two connection records differed by a single character or digit, they were not matched. As **Table 5.7** shows, all of the labelled flow (XML) files contained duplicate records, apart from the **TestbedThuJun17-*.xml** file (17/Jun/2010), while the **TestbedSunJun13Flows.xml** file (13/Jun/2010) was the worst file in terms of duplication, as it had no unique record. All duplicates were removed before the matching process took place.

Problem 4: Mismatched connection label counts between paper^[14] and XML files

In an attempt to validate the total number of labels, a comparison was made between the number of classes (i.e. the number of connections that had been classified as **Normal** as opposed to **Attack**) for every simulation day in the labelled flow (XML) files, and those presented in the

Simulation day	Number of connections	Number of copies
12/Jun/2010	133,191	1
	1	2
13/Jun/2010	136,584	2
	550	4
	24	6
	2	8
14/Jun/2010	171,322	1
	26	2
	2	3
15/Jun/2010	570,744	1
	474	2
	2	3
16/Jun/2010	522,241	1
	8	2
	2	3
17/Jun/2010	397,595	1

Table 5.7: Number of duplicate connections

authors' paper [14]. **Table 5.8** (see columns Paper and XML) shows the number of mismatched counts (in red italic), although the total number of connections (Normal+Attack) adds up.

Table 5.8 also presents the number of labels in the transformed dataset (Bro columns), based on the transformation view of connections. It shows that the number of attack connections in the **Saturday** and **Monday** traffic files were greater than those in the authored paper and XML. An investigation into this issue revealed that this was due to the fact that the XML files had bundled more packets into connections than they were supposed to. These packets should have been split into more connections as they were not complete connections. However, it seems that whatever tool the authors of the ISCX2012 dataset used did not follow network standards and that has produced the opposite effect to the split connections problem as discussed below under **Problem 5**. For example, a comparison of the attack connections in Monday's (14/Jun/2010) traffic between Bro and XML, and the number of packets involved in these connections, revealed that Bro identified fewer packets than the XML file. In Bro's results, out of 6,422 attack connections there were a total of 141,926 TCP packets, while in the XML file there were

165,755 packets in 3,721 TCP attack connections. Another reason for this phenomenon is the fact that some of the ICMP connections in the XML files aggregated exchanged sequences of ICMP packets between two hosts, whereas the Bro script used to process the PCAP files treated every ICMP packet as a single connection.

		Tag Count					
		Paper ^[14]		XML		Bro	
		Normal	Attack	Normal	Attack	Normal	Attack
Fri	11 JUN	378,667	0				
Sat	12 JUN	<i>131,111</i>	<i>2,082</i>	<i>131,107</i>	<i>2,086</i>	164,545	2,123
Sun	13 JUN	255,170	20,358	255,170	20,358	168,947	10,037
Mon	14 JUN	<i>167,609</i>	<i>3,771</i>	<i>167,604</i>	<i>3,776</i>	213,798	6,422
Tue	15 JUN	<i>534,320</i>	<i>37,378</i>	<i>534,238</i>	<i>37,460</i>	633,388	35,260
Wed	16 JUN	<i>522,263</i>	<i>0</i>	<i>522,252</i>	<i>11</i>	600,017	11
Thu	17 JUN	<i>392,392</i>	<i>5,203</i>	<i>392,376</i>	<i>5,219</i>	409,090	4,959

Table 5.8: Comparison of number of labels between authored paper, XML files, and PCAP files

Extending this analysis to the description provided on the dataset’s website [348], the description was found to be misleading. **Table 5.1** summarises this description, which reported the traffic of days, Sat (12/Jun/2010) and Wed (16/Jun/2010), as “*Normal Activity. No malicious activity*”. However, the published paper and the XML files showed contradictory figures.

With all these discrepancies it was apparent that the labelled flows (XML) reflected the ground truth of the traffic and the labels provided were used to tag the transformed dataset.

Problem 5: Split connections

Even after removing duplicate connections from the XML files, the process of matching PCAP and XML connections was not consistent. A further analysis revealed that there were multiple connections in the XML files that should have been grouped as a single connection. For example, in **TestbedTueJun15-*.xml** (15/Jun/2010) the 28 connections set out in **Table 5.9** initially formed only one connection. This was confirmed by testing the PCAP file for the simulation day on BRO [16], TSHARK [349] and TCPTRACE [350] tools, where all of these

had treated the packets exchanged between those hosts with those port numbers (192.168.2.110:3311 and 192.168.2.112:6667) as a single connection.

2010-06-15T15:53:17	2010-06-15T15:54:52	tcp_ip	192.168.2.110	3311	192.168.2.112	6667	IRC	9	726	10	2186	Attack
2010-06-15T15:56:30	2010-06-15T15:56:31	tcp_ip	192.168.2.112	6667	192.168.2.110	3311	IRC	2	144	1	79	Attack
2010-06-15T15:58:07	2010-06-15T15:59:44	tcp_ip	192.168.2.112	6667	192.168.2.110	3311	IRC	5	397	3	222	Attack
2010-06-15T16:01:19	2010-06-15T16:01:55	tcp_ip	192.168.2.112	6667	192.168.2.110	3311	IRC	10	1096	8	549	Attack
2010-06-15T16:03:21	2010-06-15T16:04:47	tcp_ip	192.168.2.112	6667	192.168.2.110	3311	IRC	9	1070	8	558	Attack
2010-06-15T16:06:18	2010-06-15T16:07:56	tcp_ip	192.168.2.112	6667	192.168.2.110	3311	IRC	4	288	2	158	Attack
2010-06-15T16:09:32	2010-06-15T16:09:32	tcp_ip	192.168.2.112	6667	192.168.2.110	3311	IRC	2	144	1	79	Attack
2010-06-15T16:11:10	2010-06-15T16:12:47	tcp_ip	192.168.2.112	6667	192.168.2.110	3311	IRC	4	288	2	158	Attack
2010-06-15T16:14:24	2010-06-15T16:14:24	tcp_ip	192.168.2.112	6667	192.168.2.110	3311	IRC	2	144	1	79	Attack
2010-06-15T16:16:01	2010-06-15T16:17:38	tcp_ip	192.168.2.112	6667	192.168.2.110	3311	IRC	4	288	2	158	Attack
2010-06-15T16:19:15	2010-06-15T16:20:51	tcp_ip	192.168.2.112	6667	192.168.2.110	3311	IRC	4	288	2	158	Attack
2010-06-15T16:22:27	2010-06-15T16:22:27	tcp_ip	192.168.2.112	6667	192.168.2.110	3311	IRC	2	144	1	79	Attack
2010-06-15T16:24:03	2010-06-15T16:25:39	tcp_ip	192.168.2.112	6667	192.168.2.110	3311	IRC	4	288	2	158	Attack
2010-06-15T16:27:16	2010-06-15T16:28:53	tcp_ip	192.168.2.112	6667	192.168.2.110	3311	IRC	4	288	2	158	Attack
2010-06-15T16:30:30	2010-06-15T16:30:30	tcp_ip	192.168.2.112	6667	192.168.2.110	3311	IRC	2	144	1	79	Attack
2010-06-15T16:32:06	2010-06-15T16:33:43	tcp_ip	192.168.2.112	6667	192.168.2.110	3311	IRC	4	288	2	158	Attack
2010-06-15T16:35:19	2010-06-15T16:36:56	tcp_ip	192.168.2.112	6667	192.168.2.110	3311	IRC	4	288	2	158	Attack
2010-06-15T16:38:31	2010-06-15T16:38:32	tcp_ip	192.168.2.112	6667	192.168.2.110	3311	IRC	2	144	1	79	Attack
2010-06-15T16:40:08	2010-06-15T16:41:43	tcp_ip	192.168.2.112	6667	192.168.2.110	3311	IRC	4	288	2	158	Attack
2010-06-15T16:43:21	2010-06-15T16:44:57	tcp_ip	192.168.2.112	6667	192.168.2.110	3311	IRC	4	288	2	158	Attack
2010-06-15T16:46:34	2010-06-15T16:46:34	tcp_ip	192.168.2.112	6667	192.168.2.110	3311	IRC	2	144	1	79	Attack
2010-06-15T16:48:10	2010-06-15T16:49:46	tcp_ip	192.168.2.112	6667	192.168.2.110	3311	IRC	4	288	2	158	Attack
2010-06-15T16:51:22	2010-06-15T16:53:00	tcp_ip	192.168.2.112	6667	192.168.2.110	3311	IRC	4	288	2	158	Attack
2010-06-15T16:54:35	2010-06-15T16:54:35	tcp_ip	192.168.2.112	6667	192.168.2.110	3311	IRC	2	144	1	79	Attack
2010-06-15T16:56:11	2010-06-15T16:57:48	tcp_ip	192.168.2.112	6667	192.168.2.110	3311	IRC	4	288	2	158	Attack
2010-06-15T16:59:24	2010-06-15T17:01:00	tcp_ip	192.168.2.112	6667	192.168.2.110	3311	IRC	4	288	2	158	Attack
2010-06-15T17:02:37	2010-06-15T17:02:37	tcp_ip	192.168.2.112	6667	192.168.2.110	3311	IRC	2	144	1	79	Attack
2010-06-15T17:04:13	2010-06-15T17:06:45	tcp_ip	192.168.2.112	6667	192.168.2.110	3311	IRC	9	1260	7	495	Attack

Table 5.9: Example of a split connection

As can be seen from **Table 5.9**, the gap between every two connections is around 90 seconds. It seems that whatever tool the authors used to analyse these PCAP files timed out any TCP connection after 90 seconds or so, which is an unreasonable timeout period for a TCP connection. Therefore, in order to address this problem, the number of connections for every two hosts in the XML files was compared with their respective PCAP files. Any difference in the number of connections triggered the aggregation process, which grouped these XML split connections into a single connection.

Problem 6: Connection with non-existing IP addresses

While it was acceptable to find a connection in a PCAP file but not in an XML file, finding a connection in an XML file but not in a PCAP file raised concerns about the reliability of the XML files. Therefore, during the validation stage of the transformation process, the packet directions and their counts were reconstructed using the relevant fields from the labelled flow (XML) files. The packet directions were then compared with those in the PCAP files (which had been pre-processed earlier and are shown in **Table 5.5**) to identify all of the connections with non-existing IP addresses, as illustrated by **Listing 5.2**.

Algorithm: Packets counts of PCAP and XML connections**Input:** pcap.packets, xml.connections**Result:** Packet counts comparison between PCAP and XML

```

1  pcktCounts <- {}
2  for (pckt in pcap.packets) do
3    key <- [pckt_proto, pckt_src_IP, pckt_dst_ip]
4    if(pckt_proto == "tcp" || pckt_proto == "udp")
5      key <- [pckt_proto, pckt_src_IP:pckt_src_port, pckt_dst_IP:pckt_dst_port]
6
7    pcap.pcktCounts[key] <- pcap.pcktCounts[key] + 1
8  done
9
10 xml.pcktCounts <- {}
11 for (conn in xml.connections) do
12   key1 <- {conn_proto, conn_src_IP, conn_dst_ip}
13   key2 <- {conn_proto, conn_dst_ip, conn_src_IP}
14   if(conn_proto == "tcp" || conn_proto == "udp") do
15     key1 <- {conn_proto, conn_src_IP:conn_src_port, conn_dst_ip:conn_dst_port}
16     key2 <- {conn_proto, conn_dst_ip:conn_dst_port, conn_src_IP:conn_src_port}
17   done
18
19   xml.pcktCounts[key1] <- xml.pcktCounts[key1] + conn_src_packets
20   xml.pcktCounts[key2] <- xml.pcktCounts[key2] + conn_dst_packets
21 done
22
23 Compare packets counts between pcap.packets and xml.pcktCounts

```

Listing 5.2: Pseudo code of comparing number of packets between PCAP and XML connections

This analysis filtered out many connections, including all IPv6 flows. For example, **Table 5.10** presents part of two records as they appeared in the labelled flow (XML) files.

...	protocolName	src_ip	src_port	dst_ip	dst_port	...	src_pkts	...	dst_pkts	...
...	tcp_ip	192.168.1.104	22441	216.246.64.49	80	...	17	...	11	...
...	tcp_ip	192.168.1.104	22445	216.246.64.66	80	...	108	...	99	...

Table 5.10: Example of XML connections

These two connections were translated into the following format as set out in **Table 5.11** so they could be matched with PCAP records for that simulation day. This was done in order to determine whether a similar packet direction existed in the PCAP files, and if it did, to check whether the number of packets matched.

ip version	protocol	source_ip	source port	destination_ip	destination port	packet count
4	tcp	192.168.1.104	22441	216.246.64.49	80	17
4	tcp	216.246.64.49	80	192.168.1.104	22441	11
4	tcp	192.168.1.104	22445	216.246.64.66	80	108
4	tcp	216.246.64.66	80	192.168.1.104	22445	99

Table 5.11: Example of number of XML packet for each communication direction

Table 5.12 presents the total number of connections for the labelled flow (XML) files for every simulation day in which at least one of its addresses or ports did not match any of those in the PCAP file. These totals also include all IPv6 connections.

Simulation day	Number of connections
12/Jun/2010	654
13/Jun/2010	564
14/Jun/2010	475
15/Jun/2010	360
16/Jun/2010	442
17/Jun/2010	1,897

Table 5.12: Number of non-matched flows in the XML Files

A further analysis was performed to identify the number of matching and missing IP:PORT combinations between the PCAP and XML files. **Table 5.13** presents the results of this analysis. The PCAP/XML cells show the number of unique IP:PORT combination that were present in both files. The PCAP/-XML cells show the number of unique IP:PORTs that did not exist in the XML files but were present in the PCAP files and *vice versa* for the -PCAP/XML cells.

			XML	-XML
Sat	12 Jun	PCAP	96,572	10,557
		-PCAP	324	-
Sun	13 Jun	PCAP	83,955	17,767
		-PCAP	52	-
Mon	14 Jun	PCAP	118,637	12,659
		-PCAP	556	-
Tue	15 Jun	PCAP	181,373	10,847
		-PCAP	2,035	-
Wed	16 Jun	PCAP	170,840	7,578
		-PCAP	170	-
Thu	17 Jun	PCAP	139,160	6,201
		-PCAP	311	-

Table 5.13: Number of unique IP:PORTs in the PCAP and XML files

Some attack connections fell into this latter category which made labelling these connections difficult. To address this problem, a partial matching strategy was used. This strategy tested all non-matched XML connections with all non-matched PCAP connections. If any two

connections partially matched then they were flagged as a possible match for manual checking. A few cases were mapped manually as only attack connections were used for the matching process and most of their keys either mapped fully or not at all.

Problem 7: Connections with wrong directions

Further analysis revealed that some connections in the XML files had the wrong direction. For example, in the labelled flows **TestbedSunJun13Flows.xml** file (13/Jun/2010), the duplicate connections set out in **Table 5.14** were manually investigated.

<u>startDateTime</u>	<u>stopDateTime</u>	<u>duration</u>	<u>protocolName</u>	<u>src_ip</u>	<u>src_port</u>	<u>dst_ip</u>	<u>dst_port</u>	<u>appName</u>	<u>src_pkt</u>	<u>src_byte</u>	<u>dst_pkt</u>	<u>dst_byte</u>
2010-06-13T16:37:36	2010-06-13T16:38:39	63	udp_ip	0.0.0.0	546	0.0.0.0	547	Unknown_UDP	0	0	7	1085
2010-06-13T16:37:36	2010-06-13T16:38:39	63	udp_ip	0.0.0.0	546	0.0.0.0	547	Unknown_UDP	0	0	7	1085

Table 5.14: Example of connections with the wrong direction

By treating these two flows as duplicates and as IPv6 connections, the communication direction can be represented as follows:

- $0.0.0.0, 546 \rightarrow 0.0.0.0, 547$, 0 packets
- $0.0.0.0, 547 \rightarrow 0.0.0.0, 546$, 7 packets

Searching for such patterns in the packet directions of the PCAP files yielded the following results:

- Pattern “**udp, .*, 547, .*, 546**”: returned no results, even though the flow record showed there were 7 packets sent in this direction,
- Pattern “**udp, .*, 546, .*, 547**”: returned the record shown in **Table 5.15**,

<u>ip version</u>	<u>protocol</u>	<u>source_ip</u>	<u>source_port</u>	<u>destination_ip</u>	<u>destination_port</u>	<u>packets count</u>
6	udp	fe80:0:0:0:5c29:ff4:d3e7:b80d	546	ff02:0:0:0:0:0:1:2	547	1183

Table 5.15: Example of a UDP Connection's wrong direction

These results showed a mismatch between the total number of packets exchanged. They also raised concerns about setting the source port for the flow as 546 and the destination port as 547, when all traffic direction for this flow should have been the other way around.

The main problem in such cases was the extra complication involved in the validation. These problems might have raised lots of inconsistencies, even if the traffic did not affect the actual labelling process, as they did not exist in the attack connections.

Problem 8: Connection with source packets ZERO

The example discussed under **Problem 7** led to further analysis to identify all the flows with zero source packet counts. **Table 5.16** presents the total number of such connections for each simulation day. This issue was another reason that packet counts were not used at the labelling stage to match the connections between the PCAP and the XML flows.

Simulation day	Number of connections with ZERO source packets
12/Jun/2010	377
13/Jun/2010	706
14/Jun/2010	438
15/Jun/2010	874
16/Jun/2010	1,063
17/Jun/2010	1,481

Table 5.16: Number of connections with zero source packets

Problem 9: Connection with wrong source or destination packet

Further analysis was undertaken to identify any connections where the total number of packets exchanged did not match those in the PCAP files (**Table 5.17**). During this analysis the packet directions were reconstructed from the labelled flow (XML) files and compared with the PCAP files (see **Listing 5.2**). All flows used to reconstruct those packet directions that did not match the number of packets in the PCAP files were flagged up.

Simulation day	Number of connections with the wrong number of packets
12/Jun/2010	21,428
13/Jun/2010	272,014
14/Jun/2010	22,813
15/Jun/2010	67,439
16/Jun/2010	93,297
17/Jun/2010	45,629

Table 5.17: Number of connections with the wrong number of packets

This mismatch in the number of packets made the process of mapping connections between PCAP and XML files difficult and would have raised doubts about the validity of the resultant dataset if it was not explained here.

5.4 Server Specifications

This experiment was run using a server with 2U Supermicro chassis; 8x host-swap 2.5" SAS/SATA disk bays; Supermicro X8DTU-LN4F+ motherboard; Dual Intel Xeon E5620 (quad core) ; 24GB RAM (6 x 4GB DDR3 ECC RDIMM) ; 4x 1TB SATA (RAID10) and 4x 1Gb Ethernet. It used a Windows Server 2012 R2 Datacentre (64-bit) Operating System [351].

A virtual machine (VM) was created on Hyper-V with 4 Virtual Processors and 4 GB RAM. This VM was used to host the SecurityOnion (12.04.5.1-20150205) operating system [352], which had Bro (2.4), Perl (5.18.2) [353], TCPTRACE (6.6.7) [350] and TShark (1.6.7) [349] installed to run these experiments. Bro was chosen to process the PCAP files because of its high-speed, extensibility and ability to extract features at multiple levels (frame header, IP header and transport headers) at the same time. It also had the capabilities to extract the content of unencrypted traffic if needed. TShark and TCPTRACE were used to validate the results of Bro's TCP connections, which provided confirmation that the numbers in the generated dataset added up for all the targeted traffic. Perl was used to map the processed connections to their correct labels in the traffic flow provided by the ISCX2012 dataset. "DMwR" package

(0.4.1) [346] in R software [321] was used to run the SMOTE algorithm [207] to generate the synthetic attack traffic.

5.5 Limitations

The transformation process discussed above produced a large dataset in the network security domain that addressed many of the limitations of other known datasets in the field. However, there were still a number of limitations to the process that was used.

Firstly, even though the ISCX2012 dataset contained all of the exchanged payload unencrypted, this transformation did not generate any content based features that were similar to those in KDD [155], NSL-KDD [157] or gureKDD [160-162]. The decision to make the generation process generic was taken to avoid any complications in a real life environment, such as encryption and privacy concerns as explained in **Section 5.2.1**. In addition, another reason was the fact that the payload of every service required a different set of features specific to that service, so producing a general set of features to profile the content of all the different services would have been challenging. Moreover, this kind of profiling could be addressed through another line of research, where service specific IDS could be investigated.

Secondly, this transformation adopted the settings suggested by Lee *et al.* [354-356], Stolfo *et al.* [155] and Perona *et al.* [162] in using a window of 100 connections to derive the connection-based features and followed the documentation of Onut *et al.* [15] in using a window of 5 seconds for the time-based features. It is not clear if these sizes are for all network traffic and targeted profiling, or whether they should be adapted to set the right window size for specific traffic. Further investigation is required to analyse the effect of different connections and different time window sizes on different traffic patterns and attack types.

Thirdly, the SMOTE algorithm was used to generate synthetic traffic and balance the dataset. However, it is not clear if the SMOTE algorithm was the best choice for a such domain, i.e.

network security, as the values of a connection feature are not actually random because of the standards governed by networking protocols. Any instances generated by the SMOTE algorithm which introduced randomness could cast doubt on the validity of those samples. Therefore, as this issue might have affected the quality of the dataset, the generation process ensured that every connection was identifiable. Researchers can omit these synthetic instances, or even use the original connections in the dataset, to come up with their own balanced version using whatever technique best fits their research aim.

Fourthly, the labelling of this dataset used the tags provided within the XML files which only provide binary options (**Normal** or **Attack**). Although, different attack scenarios were performed in the ISCX2012, further investigation would have been required to distinguish them. Due to time limitations, the binary labels provided by the flow (XML) files were used.

Finally, this transformation process assumed that all connections in the dataset were normal except for those labelled otherwise in the XML files. This decision was dictated by the issues already discussed in relation to the ISCX2012 dataset which effected the mapping and labelling process. For example, a processed connection (from the transformation process) could have been mapped to a number of split connections from the XML files (as discussed in **Problem 5** in **Section 5.3.3**), where the split connections had mixed labels. In such cases **Attack** was used to label these connections in the resultant dataset.

5.6 Summary

This chapter has outlined the transformation process of the UNB ISCX 2012 dataset into a KDD-like format. This transformation took into account many of the lessons learned during the analysis of the KDD 1999 dataset [1] and the investigation of the flow files in the ISCX2012 dataset. These lessons could be summarised as guidelines for dataset generators and authors who require a comparable transformation as follows:

1. **Capture every targeted packet.** This requirement ensures that no packet is dropped or neglected without valid reason. In any transformation, the total number of packets in the original (raw) dataset should match the resultant (transformed) dataset.
2. **All values** (the number of packets, bytes sizes, durations, timestamps, *etc.*) **should be computed correctly.** This is to address the limitation of using nonstandard tools to perform packet processing. For this reason, this experiment used well-known software (Bro) for flow processing. Another possible obstacle is the misconfiguration by the tool used, which could overlook some traffic which would in turn result in a loss of information. Also the total number of packets in the PCAP files must be equal to the sum of packets for all profiled connections. With this overview any mismatches can then be investigated.
3. **Correctly extract every IP address.** This is similar to all IPv6 addresses in labelled flow (XML) files in the ISCX2012 dataset not being processed correctly, resulting in useless data.
4. Use **timestamps** rather than the human readable date/time format, (or use both). This is because the human readable date/time format will not translate accurately or with the precision of the original timestamp when converted back to match connections.
5. Use **multiple standard tools** or libraries to ensure the same view of connections and to provide guidance when any differences arise (TCPTRACE, vs tShark, vs BRO, *etc.*) between tools. Differences usually arise when a tool is configured in such a way that it is not readily accepted in production environments; this could be picked up when different views of the same traffic are produced by multiple standard tools.
6. **Ensure that every flow direction is correctly represented.** This is to avoid any problems due to a code error or a tool bug that might mix up flow processing, and

aggregate packets in the wrong direction which would affect the quality of the transformed data.

7. **Transformation should be based on a clear definition of connections.** Ensure that every connection is clearly defined for every targeted protocol, so that the start and end criteria of such connections are clearly defined. For example, some tools will define a TCP connection from the first SYN packet to the last FIN packet, while others might define the start of a connection as being from the successful completion of the handshake phase to the last FIN packet or a certain idle period. Consistent definitions will ensure the transformation process is reproducible by other researchers.

As a result of following these guidelines, it is believed that the resultant dataset (**STA2018**) of this study provided the most accurate profile for every connection in the UNB ISCX 2012 dataset. The **STA2018** will be used for the experiments of the following chapters.

Chapter Six

Effect of Feature Selection and Data Balance on Adaptive Cutoff for Network Intrusion Detection

This chapter extends the analysis outlined in **Chapter 4** to a domain specific problem, namely, *Network Intrusion Detection*. It looks at the effect of **data balance** and **feature selection** on the performance of detection models before and after the adaptive cutoff (threshold) has been applied to the dataset being evaluated. It also considers the use of different salient features in combination with data balancing of the training data in developing prediction models using different ML algorithms (C5.0, Random Forest and SVM). The effects of various combinations of these parameters on the performance of these models (G-Mean Accuracy) in predicting traffic with previously unseen attacks, different statistical properties, and different feature sets are analysed. This analysis is based on a new dataset (STA2018) that was generated specifically for this analysis and the generation process was outlined in **Chapter 5**.

6.1 Introduction

As discussed earlier, in **Chapter 2**, feature selection plays an important role in the model generation process. However, the features selected for the model generation phase will not necessarily reflect the same important features of the evaluated data if they introduce different statistical properties. This phenomenon is widely recognised in highly dynamic environments, such as network traffic, where traffic on one day is likely to have different important features to traffic on other days due to changes in traffic patterns. This effect is not usually captured in

the conventional k-folds Cross-Validation technique when applied to the (dataset) traffic of all days collectively. That is because the data in such an approach are partitioned using random sampling, and the models are generated from a larger proportion of the full dataset, where the feature selection process has been dictated by the overall properties of the full data. In a dynamic environment, data will evolve through time, producing shifts in feature importance resulting in different feature sets to those used to train the model. In the data stream domain (which is out of scope for this project), changes in feature importance is considered an important indication of concept drift [135, 138, 140, 141, 279, 339, 340], which trigger a model updating/training phase. This chapter therefore analyses the applicability of the threshold adaptation approach in reducing, or eliminating, the effects of differences between traffic patterns (concept drift) on the performances (accuracy) of different prediction models. These models were generated with different feature sets (which were selected based on the training data alone), and different data balances. Their performance was then assessed against the traffic from other days, which would have new patterns with novel attacks and different feature sets.

In the experiments discussed in this chapter, the Random Forest (RF) algorithm, first proposed by Breiman [171, 357-360], was used for the feature selection task. It is one of the most important and widely used algorithms in the ML domain, as it can perform many tasks, including classification and regression. One of the RF's strongest properties is its ability to rank features by their importance. RF supports **wrapper** methods (see **Section 2.2.3.1**) and can compute feature importance by two approaches: **Mean Decrease of Accuracy (MDA)** and **Mean Decrease in Gini (MDG)**.

6.1.1 Feature importance measures

The **Mean Decrease of Accuracy (MDA)** [297, 360] is computed by measuring the Out-Of-Bag (OOB) error for every data record in the training data. The OOB error is the average classification error for every observation, n_i , by trees that did not use this observation (n_i) in

their training process [169]. To compute the MDA of features, every feature, F_i , in the training dataset is shuffled and the OOB error is recalculated. The average error of difference, which is normalised by these differences' standard deviation between pre and post shuffling, is used as an importance score, whereby the higher the error, the higher the importance [297, 360]. In other words, this measure represents the number, or the proportion, of instances that have been misclassified as a result of removing feature F_i from the model. This process can take a long time to execute, depending on the amount of data, the number of features, and the number of trees in the forest.

The **Mean Decrease in Gini (MDG)** [297, 360] impurity is another metric used by RF to rank features; it basically measures the probability of misclassifying (mislabelling) a randomly selected instance from a set where the set's label distribution has been used to classify that instance. Classification trees usually use this metric (Gini impurity) to decide which feature should be used to perform the split in the fitted tree. In other words, for every feature, F_i , the Gini impurity (gain of purity) measure is computed, and the feature with the best (or highest) measure is used to split the tree. Simply put, the usefulness of feature, F_i , is evaluated based on its ability to split mixed (impure) nodes into single label (pure) nodes. Therefore, the decrease in node impurity (or increase in purity) for every feature, F_i , that is used in performing tree splits is averaged out over all the trees in the forest. Finally, features were ranked using this measure, so that those with a higher decrease in node impurity are regarded as having higher importance.

Computation using MDG was much faster compared to using the MDA. However, the computation of feature importance using the MDG method is fast and it presents the average of purity gain at local splits. As a result, this measure (MDG) may not be good at selecting features which are able to generalise to different test and evaluation data, as they are more inconsistent and biased. Also, unlike MDA, it is not linked to the model's performance. Despite these issues,

MDA and MDG were used as the core evaluation methods in the feature selection task in this study.

As with many other feature selection methods, the RF ranks the returned features by their importance measure, where some selection criteria is performed to set the cut-off point in these rankings to select the features subset. Deciding where to set this cut-off or how many variables to use - for the final model generation - needs to be subjected to iterative testing to evaluate different models with different selected feature sets. As this task would have consumed precious time, an alternative method was applied.

6.1.2 Feature selection using fake features

Bi *et al.* [361] have attempted feature selection through introducing a probe to the data by adding three randomly generated variables (fake features/columns) to the dataset. These fake features are randomly drawn from a Gaussian distribution [188]. They use a linear SVM to model the subsets at every iteration of a K-folds Cross-Validation, where variables with nonzero weights are selected. Any variable (feature) with an average weight below that of the fake variables is then rejected. This approach does not address weight variability as it only compares averages.

Similarly, Kurasa *et al.* [362, 363] have proposed a similar approach in which the information system (training data) is doubled, so that every feature has a shadow feature that is basically a shuffled version of the original one. Feature importance evaluation is then performed on the extended system using the RF algorithm. A K-folds Cross-Validation - of at least 10-folds - is performed at every iteration so that every feature is compared to its shadow using statistical tests to evaluate the highest performing features. The main drawbacks of this approach are scalability and speed. Therefore, in this chapter a new approach has been proposed and executed which combines the core ideas of the two approaches above.

In this approach, as illustrated in **Figure 6.1**, the information system (training data) is extended by adding three randomly generated variables (fake features/columns) to the dataset, where these fake variables are drawn randomly from a Gaussian distribution. A feature importance evaluation - using the RF algorithm - was performed on the newly extended system and the importance measures of these random variables were then used as a threshold to reject any features with a lower importance value than those of the fake variables. In other words, any feature that performed worse than a random guess was rejected. This comparison was performed using statistical measures.

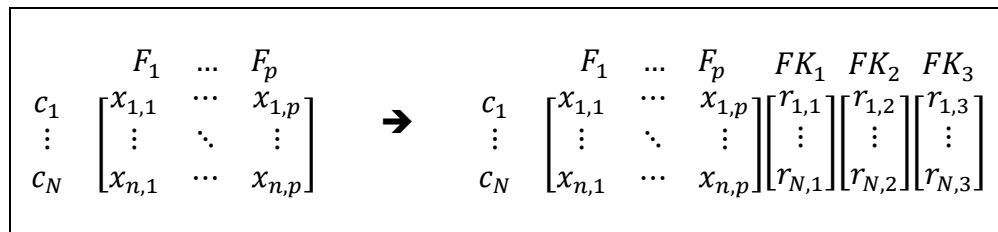


Figure 6.1: Illustration of the information system extension with fake features/variable.

As equal variance between compared groups (feature versus fake variables) is not guaranteed, and due to the unbalanced design (number of compared importance measures) of these comparisons, which would have small sample sizes, Welch's two sample t-test [364, 365] was used. Comparisons were performed to evaluate the statistical significance of the mean difference between every feature and the fake variables. The aim of this approach was to speed up the feature selection stage and to make it independent of human evaluation or fixed thresholds so that it would be more adaptive to the true nature of the dataset. This study adapts the approach of Bi *et al.* [361] to address the limitation of the Kursu *et al.* [362] method.

Every fake feature was formed of N random values drawn from a Gaussian distribution with a mean of zero and a standard deviation of one, where N was the number of records in the training data. These random features were combined with the original dataset and were processed by the RF algorithm to compute its features importance, using a 3-folds Cross-Validation technique. A **Welch's t-test** statistic comparison was then performed to evaluate whether the

mean of the importance measures of every feature, F_i , - from the three folds - was statistically significantly greater than the mean importance of the fake features (with a significance level of $\alpha = 0.05$). All features with a mean importance statistically significantly greater than that of the fake features were selected. The steps of the feature selection stage are illustrated in **Listing 6.1**. The Ranger package [297], which is a fast RF implementation in C++, in the R software [321] was used for these experiments.

In the feature importance evaluation, 15 categorical (factor) features were eliminated, as they had been added to all the experiments' model building designs and evaluation process by default. These features are listed in **Table 6.1**.

No.	Feature	No.	Feature
2	src_ip	24	bro_service
3	src_zone	31	conn_start
5	dst_ip	32	conn_partial_start
6	dst_zone	33	conn_close
9	ipVersion	34	conn_partial_close
10	Protocol	43	conn_stats_orig_endian_type
11	conn_state	50	conn_stats_resp_endian_type
23	bro_conn_state		

Table 6.1: Categorical (factor) features eliminated from the feature importance evaluation phase.

6.1.3 Data balance

Data imbalance is another important issue that affects ML and data analytic tasks. This issue appears when the data used have more samples of one class than another (binary classification) - known as majority and minority classes - or non-equal proportions of classes in a multi-classification problem. This is a well-known issue in domains such as network ID, where normal connections are far greater than anomalous traffic [366]. This problem affects some classification methods which perform poorly on minority classes, due to the inability of these algorithms to detect the distribution of the relative class [367]. A number of methods have been proposed to address this issue, such as undersampling and oversampling [223, 368, 369] (see **Section 2.2.3.2**).

Algorithm: Feature Selection with Fake Features	
Input:	dataFile, ftrType
Result:	Selected Important Features
1	dataFile <- filename, // Name of the data file to be processed
2	ftrType <- ftrMsr, // Features importance measure {MDA or MDG}
3	
4	ftrImportance <- {}, // Initialize list to contain the computed
5	// importance value of every feature
6	ftrSelected <- {}, // Initialize list to contain the selected features
7	
8	DS <- load file (fileName), // Load the content of the data file
9	ftrSet <- getDataFeatures(DS), // Get the list of features in the data file
10	N <- num_rows(DS), // Get number of records in the training data
11	
12	FK ¹ <- rand(sample=N, mean=0, sd=1), // Generate 3 lists of random variables where
13	FK ² <- rand(sample=N, mean=0, sd=1), // each list contains N random numbers with
14	FK ³ <- rand(sample=N, mean=0, sd=1), // mean=0 and standard deviation=1
15	
16	
17	newDS <- [DS _(N×p) FK ¹ _(N×1) FK ² _(N×1) FK ³ _(N×1)], // Append the fake features to the original data
18	partsDS <- create K partitions of newDS, // Create K partitions to calculate features
19	// importance measures using K-folds Cross-Validation
20	
21	// Compute the importance of every feature using K-folds
22	// Cross-Validation and save them in ftrImportance
23	For fold in K-folds, do
24	trainRcrds <- partsDS[-c(fold)]
25	ftrImportance[fold,] <- featre_importance(data=newDS[trainRcrds,], measure=ftrMsr)
26	done
27	
28	// Evaluate every feature in the data file by comparing its performance
29	// to the performances of the 3 fake features. If the mean importance of
30	// that feature is statistically higher than the mean importance of the
31	// fake features, then add that feature to the selection set.
32	For F _i in ftrSet, do
33	if(ftrImportance[,F _i] > ftrImportance[,c(FK ¹ , FK ² , FK ³)] with t.test probability > 0.05){
34	ftrSelected <- ftrSelected U {F _i },
35	}
36	
37	done
38	
39	return(ftrSelected), // Return the list of selected features

Listing 6.1: Pseudo code of the feature selection function.

The experiments outlined in this chapter employed an oversampling technique and adopted the SMOTE technique - as discussed in **Chapter 5** - to balance the datasets.

6.2 Proposed Solution

This chapter investigates the potential of the adaptive cutoff (threshold) approach in a setup similar to a real life scenario. In this setup, different binary predictive models were created using a subset (one day's traffic) of the dataset with different feature sets (Full, MDA, MDG, MDA_{Balance} and MDG_{Balance}) and different data balances (Original and Balanced). Each of these models were then used to predict traffic on the other days. These experiments aimed to analyse

the extent to which the cutoff adaptation technique improved the performance of different predictive models.

These models were generated using features that differed from those of the evaluated data. The following is an overview of the main set of features that were selected for each day of traffic using the **STA2018** dataset (see **Listing 6.1**):

- **Full** features: this set of features, which had a total of 544 features, contained all the features of the original transformed dataset (after eliminating five features as detailed in **Section 6.3**).
- **MDA** features: are the set of features selected using the feature importance measure - Mean Decrease of **Accuracy** (MDA) - returned by the Random Forest algorithm. The **original** (imbalanced) data were used in the evaluation and selection of these features.
- **MDG** features: are the set of features selected using the feature importance measure - Mean Decrease **Gini** (MDG) - returned by the Random Forest algorithm. The **original** (imbalanced) data were used in the evaluation and selection of these features.
- **MDA_{Balance}** features: are the set of features selected using the feature importance measure - Mean Decrease of **Accuracy** (MDA) - returned by the Random Forest algorithm. The **balanced** data were used in the evaluation and selection of these features.
- **MDG_{Balance}** features: are the set of features selected using the feature importance measure - Mean Decrease **Gini** (MDG) - returned by the Random Forest algorithm. The **balanced** data were used in the evaluation and selection of these features.

The number of selected features in each set of features (MDA, MDG, MDA_{Balance} and MDG_{Balance}) was bound to differ from one subset (day file) to another, as feature importance varied due to the differences in traffic patterns and statistical properties. Note that the size of

the Full features set was the same across all days. A full list of the selected features for each day is presented in **Appendix (B)**.

In these experiments three algorithms (C5.0, RF and SVM) were used to build **ten** binary predictive models each for every data file (subset) in the **STA2018** dataset. Five of these models were generated using the original (imbalanced) data file for the day with five different sets of features [Full, MDA, MDG, MDA_{Balance} and MDG_{Balance}]. The same set of features were then used to generate another five models using the balanced version of the data. All ten models were used to evaluate the remaining original (imbalanced) data files, and their performance was reported in terms of G-Mean Accuracy.

Rodríguez *et al.* [370] recommend the use of $K > 2$, in the K-folds Cross-Validation (CV) technique, for less biased estimations. However, given the large size of the datasets under consideration, larger values of K (i.e. $K=5$ or $K=10$) would have resulted into higher computational costs for every experimental run. As a result, the 3-folds CV technique was used to build and set the prediction optimal (CV) threshold for each model. At the evaluation stage, two readings of the same measure (G-Mean Accuracy) were recorded for the performance of every model; one was based on the model's optimal (CV) prediction threshold and the second on its performance after threshold (cutoff) adaptation of the evaluation (test) data. As noted earlier, these experiments aimed to mimic real life situations where training and testing datasets have different statistical properties. Further details about the setup and configuration of the experiments are presented in **Section 6.4**.

Every generated model was trained for binary classification and configured to return the probability for the class rather than for the class label. An advantage of using class probability is the flexibility that it offers in computing a model's performance at different prediction thresholds and thus determine the peak point.

These experiments aim to test the following key hypotheses to evaluate the effect of various variables:

Threshold-H₀: *“there are no statistically significant differences in model performance (G-Mean accuracies) before and after cutoff (threshold) adaptation has been applied.”*

ML-H₀: *“there are no statistically significant differences in model performance (G-Mean accuracies) between the different ML algorithms (C5.0, RF and SVM) before and after cutoff (threshold) adaptation has been applied.”*

Features-H₀: *“there are no statistically significant differences in model performance (G-Mean accuracies) between the different feature sets (Full, MDA, MDG, MDA_{Bal.} and MDG_{Bal.}) before and after cutoff (threshold) adaptation has been applied.”*

Balance-H₀: *“there are no statistically significant differences in model performance (G-Mean accuracies) between the different data balances (Original and Balanced data) before and after cutoff (threshold) adaptation has been applied.”*

The effect of threshold adaptation on the overall accuracy of the predictions of various models developed using different feature sets and data balances are further analysed by testing the following hypothesis for every ML algorithm: *“there are no statistically significant differences in the performance (G-Mean accuracies) of models built with different feature sets and different data balances after a cutoff (threshold) adaptation has been applied.”*

6.3 Datasets

All of the experiments outlined in this chapter used the **STA2018** dataset which full details of its generation are provided in **Chapter 5**. For the analyses set out below, two variations of the **STA2018** dataset were used. The first was the original data extracted from the original ISCX2012 dataset, as shown in **Table 6.2** which sets out the number of connections for each class for each day. The second version was a balanced version of the **STA2018** dataset. For this dataset, the SMOTE algorithm was used to generate synthetic instances of the attack connections (minority class) to balance the dataset. (Full details of this process are given in **Chapter 5**).

In the experiments outlined below, only days two to seven were used, as the first day was attack free.

	Day 1 11/Jun	Day 2 12/Jun	Day 3 13/Jun	Day 4 14/Jun	Day 5 15/Jun	Day 6 16/Jun	Day 7 17/Jun	Total
Normal	442,705	164,545	168,947	213,798	633,388	600,017	409,090	2,632,490
Attack	0	2,123	10,037	6,422	35,260	11	4,959	58,812
Total (Original)	442,705	166,668	178,984	220,220	668,648	600,028	414,049	2,691,302
Synthetic	0	162,422	158,910	207,376	598,128	600,006	404,131	2,130,973
Total (Balanced)	442,705	329,090	337,894	427,596	1,266,776	1,200,034	818,180	4,822,275

Table 6.2: Number of classes of instances for each day's file of the STA2018 dataset.

Originally, the file for each day consisted of 550 features (549 features + 1 class). The added features of **synthetic** and **origOrder** were omitted from the analysis as their only purpose was to distinguish the original data from the balanced (synthetic) data and to identify the connection order. Three further features were removed from the analysis (**start_time**, **src_ip** and **dst_ip**), both to avoid any possibility of overfitting and because of the large number of levels. Removing these five features resulted in a total of 545 features (544 features +1 class). Any reference to the **Full** set of features, thus refers to these 545 features.

The number of features selected for each day is presented in the shaded cells of **Table 6.3**. The cells below the diagonal separator set out the total number of shared features for each pair of days that were selected by using the **MDA** feature importance measure on the **original** data. Whereas, the upper part of the table presents the total number of shared features using the same measure on the **balanced** data. For example, using the MDA measure to compute feature importance for the original data for Day 2 resulted in the selection of 130 features, whereas using the MDA measure on the balanced version of the same data resulted in the selection of 166 features.

		Day 2	Day 3	Day 4	Day 5	Day 6	Day 7		
MDA	Day 2	166 130	164	150	154	106	105	Day 2	MDA _{Balanced}
	Day 3	126	507 518	369	369	169	310	Day 3	
	Day 4	124	352	378 364	335	158	206	Day 4	
	Day 5	125	356	327	388 368	164	208	Day 5	
	Day 6	42	59	59	58	170 60	108	Day 6	
	Day 7	91	345	224	215	41	322 355	Day 7	

Table 6.3: Number of shared selected features for each two day pair after using the Mean Decrease of Accuracy (MDA) importance measure on the original and balanced versions of the data.

The total number of shared features for Day 2 and Day 3 when the original data were used was 126, whereas 164 features were selected when the balanced data were used.

Table 6.4 presents the same comparisons but for the MDG measure. It is worth noting that the `start_time` feature was selected for most days by all measures but due to experimental design decisions this feature was subsequently removed from all model fitting and evaluations to avoid overfitting problems.

		Day 2	Day 3	Day 4	Day 5	Day 6	Day 7		
MDG	Day 2	119 124	41	37	63	46	61	Day 2	MDG _{Balanced}
	Day 3	2	137 27	60	60	36	49	Day 3	
	Day 4	1	0	117 11	60	27	41	Day 4	
	Day 5	49	1	6	168 113	58	81	Day 5	
	Day 6	61	1	1	34	84 70	44	Day 6	
	Day 7	65	12	7	43	38	134 137	Day 7	

Table 6.4: Number of shared selected features for each two day pair using the Mean Decrease Gini (MDG) importance measure on the original and balanced versions of the data.

Table 6.3 and **Table 6.4** only present the total number of common features for each two day pair. An UpSet technique is used to show the detailed intersections between different days of every set of features. This technique was used instead of a Venn diagram, as the latter fails to visualise clearly the intersections of a large number of sets.

UpSet [371, 372] is an open-source visualisation technique used to perform quantitative analyses of sets. It can present set intersections as well as aggregates of intersections. In an UpSet plot, the total number of a set's members - the selected features for each day - is presented in the lower left-hand corner, while the main plot illustrates the aggregation of intersections in one of three ways: by *degree*; by *pairwise overlap*; or by *sets*. The difference between the three is illustrated in **Figure 6.2-(a)**.

In this chapter, the aggregation of intersections are illustrated by '*degree*' i.e. they present the number of elements in every non-empty slice of the intersection between all sets in increasing order of degree. For example, in a dataset with k sets, there will be 2^k possible intersections [371, 372]. Each one of those intersections corresponds to an atomic area (slice) of the Venn diagram. Every exclusive intersection area (slice) is denoted by filled dark circles connected by a line that indicates which sets are contributing to the formation of this slice. **Figure 6.2-(b)** provides an example of all the possible ways to represent the intersections of a three set relationship.

By comparing the total number of common features (intersections) amongst days using the same measure, as presented in **Table 6.3** and **Table 6.4**, it can clearly be seen that there are more common features for different days using the MDA measure than using the MDG measure. The UpSet plots outlined in **Figure 6.3** and **Figure 6.4** reveal the same observation.

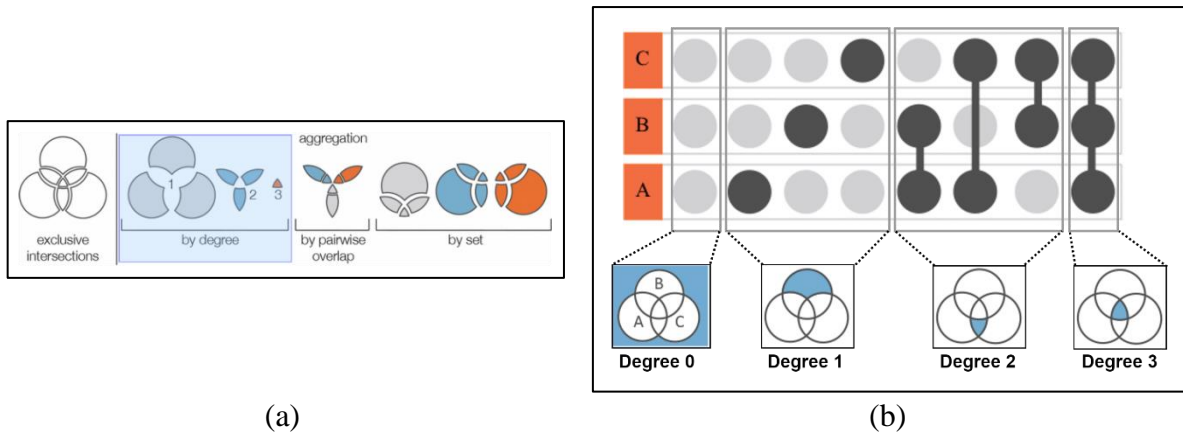
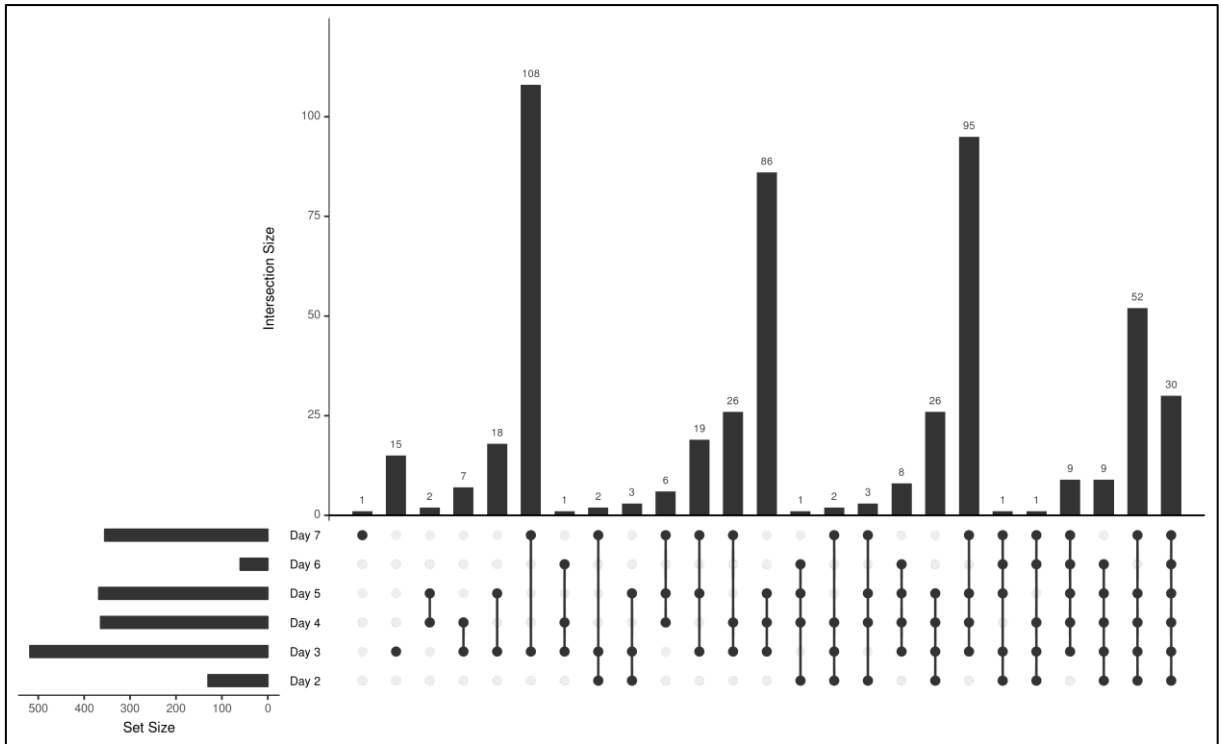
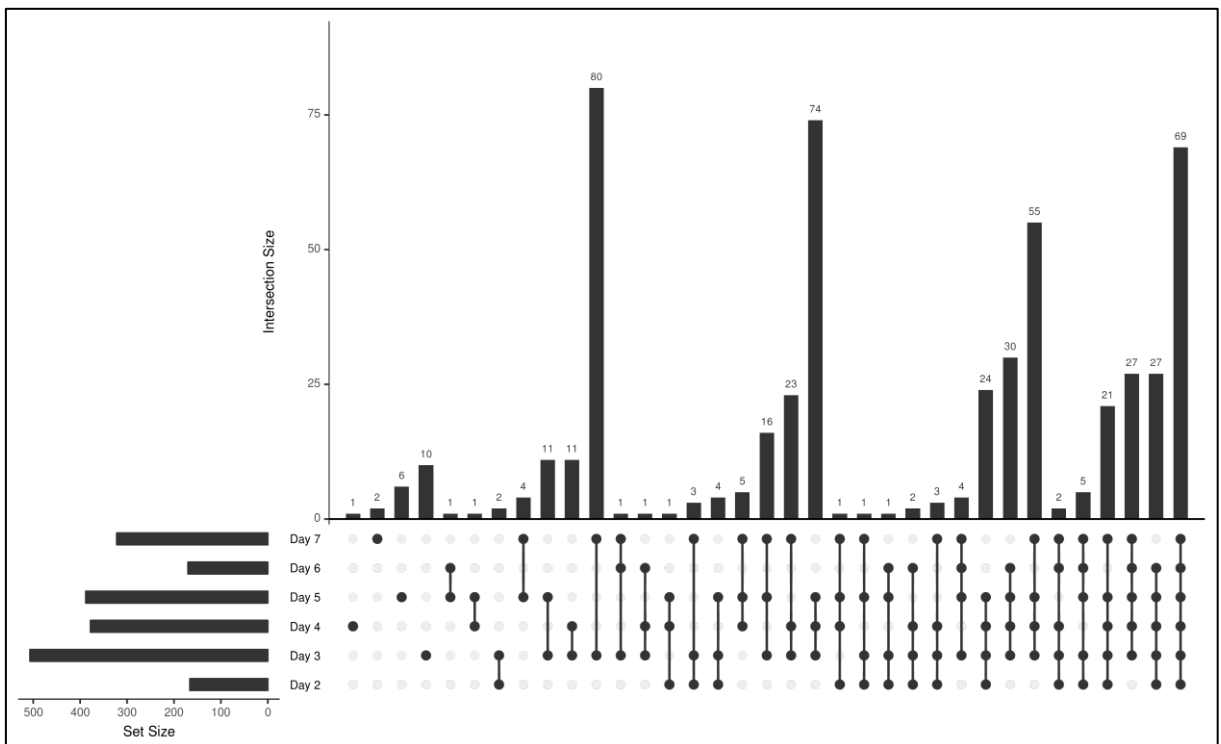


Figure 6.2: Figures adapted from Lex et al. [372]. (a) Examples of slicing and aggregation, including aggregation 'by degree' which is used in this chapter. (b) Example of a set relationship encoded by columns from the matrix, where the sets that contribute to every exclusive intersection are represented by filled dark circles connected by a line.

Figure 6.3-(a), which presents the MDA feature sets, shows that there was only one unique feature in Day 7 that did not exist in all the other groups, whereas Day 3 had 15 unique features. The figure also shows that there were a total of 30 features common between all days. The $MDA_{Bal.}$ feature set, illustrated in **Figure 6.3-(b)**, shows that balancing the data increased the number of common features for all days to 69. **Figure 6.4 (a)** and **(b)**, which presents the MDG and $MDG_{Bal.}$ features sets respectively, shows that the MDG measure resulted in more unique features for specific days than the MDA measure. In general, the MDA measure resulted in the selection of more shared features for different days than the MDG measure. Furthermore, balancing the data increased the number of selected features for each day, resulting in larger intersections. This can be clearly seen from the number of features in intersections of the same degree presented in **Table 6.5**, which summarises the plots in **Figure 6.3** and **Figure 6.4**. As can be seen from this table, the MDG and $MDG_{Bal.}$ feature sets have a higher proportion of features in intersections of degree 1, which are unique to individual days. This confirms the behaviour expected of the MDG measure in evaluating features locally as discussed in **Section 6.1**.

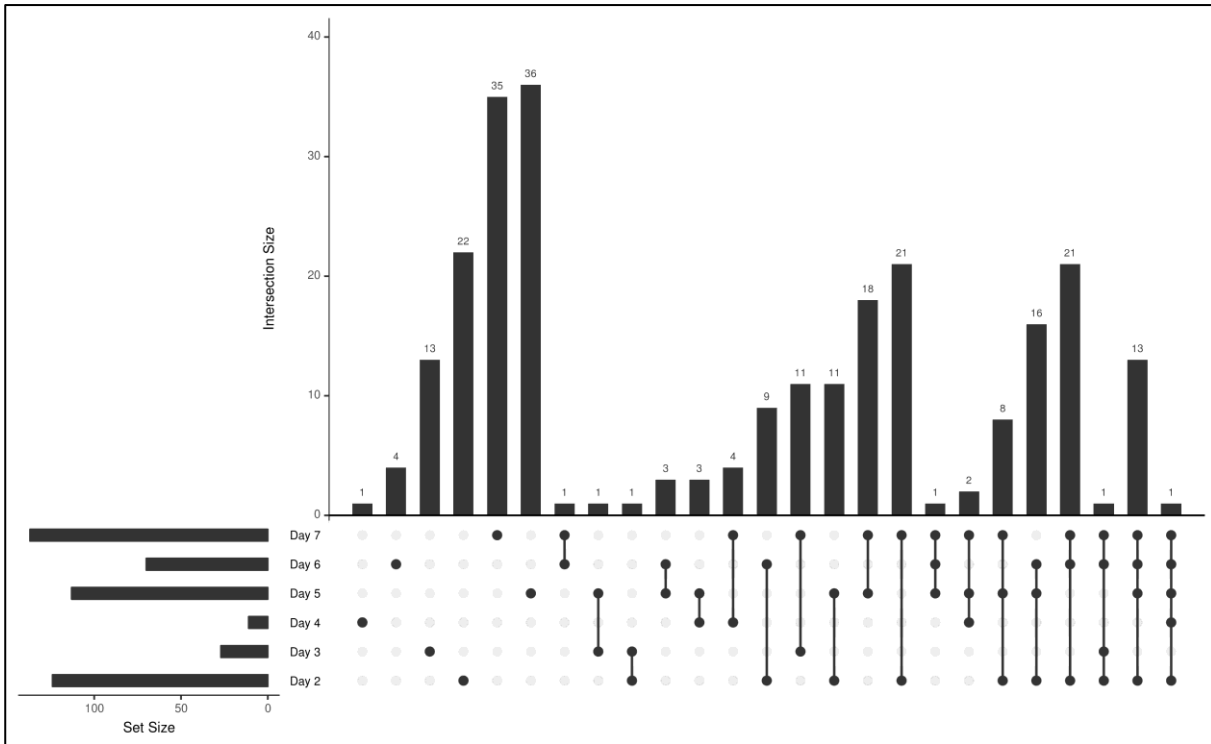


(a)

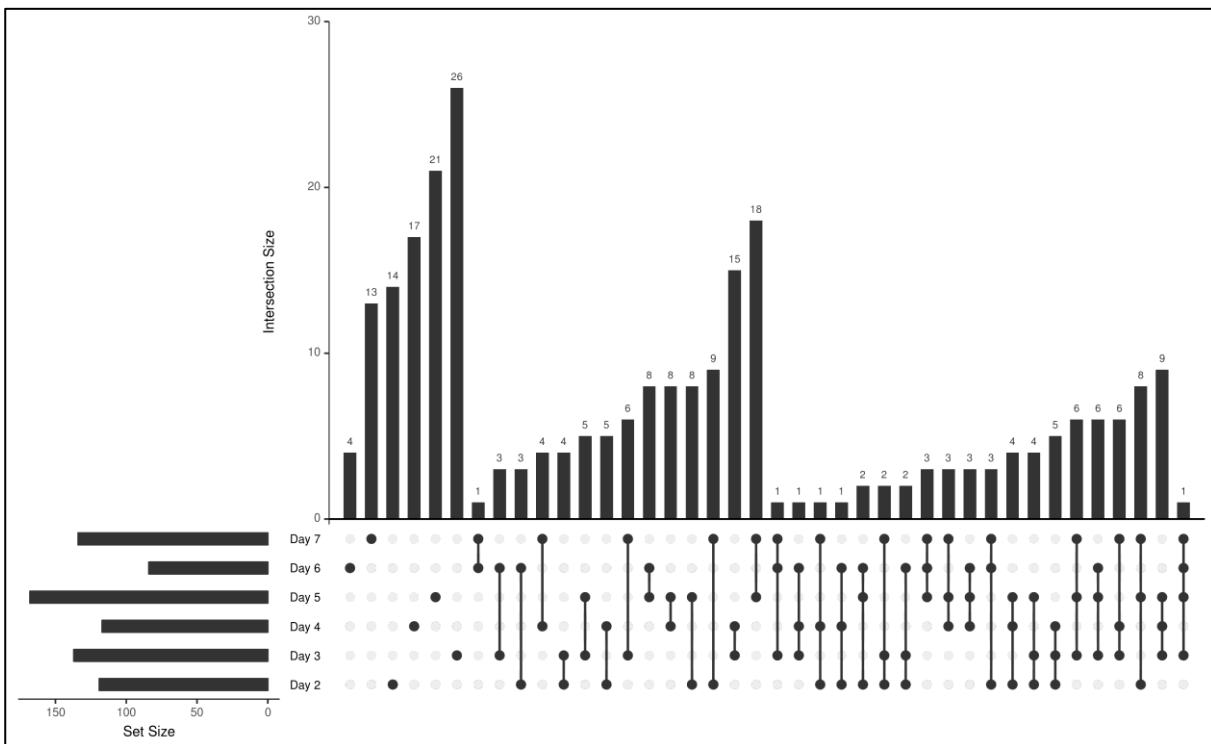


(a)

Figure 6.3: Plots of feature sets' intersections using the MDA measure (a) Day Features' intersections using the MDA measure on the original (imbalanced) data. (b) Day Features' intersections using the $MDA_{Balance}$ measure on the balanced data.



(a)



(a)

Figure 6.4: Plots of feature sets' intersections using the MDG measure (a) Day Features' intersections using the MDG measure on the original (imbalanced) data. (b) Day Features' intersections using the MDG_{Balance} measure on the balanced data.

		Features Sets			
		MDA	MDG	MDABal.	MDGBal.
Number of features	Deg. 1	16	111	19	95
	Deg. 2	135	83	110	97
	Deg. 3	143	48	128	70
	Deg. 4	135	14	121	1
	Deg. 5	72	1	82	0
	Deg. 6	30	0	69	0
Total (Proportion of Full 544 features)		531 (97.61%)	257 (47.24%)	529 (97.24%)	263 (48.35%)
Proportion of features	Deg. 1	3.01%	43.19%	3.59%	36.12%
	Deg. 2	25.42%	32.30%	20.79%	36.88%
	Deg. 3	26.93%	18.68%	24.20%	26.62%
	Deg. 4	25.42%	5.45%	22.87%	0.38%
	Deg. 5	13.56%	0.39%	15.50%	0.00%
	Deg. 6	5.65%	0.00%	13.04%	0.00%

Table 6.5: Number of features in all intersection areas (slices) of an aggregation degree for every feature set and their proportions in relation to the total number of unique features within each feature set.

6.4 Experimental Setting

The experiments discussed in this chapter were evaluated, in terms of classification performance, using the **G-Mean Accuracy** measure. The experiments were executed in **three** different phases as explained below and illustrated in **Figure 6.5**. (For greater clarification, **Listing 6.2** presents a pseudo code for these phases.)

Before performing any of the phases, a day's traffic file (subset) was pre-processed in order to balance the number of class instances in each file (*line 6* in **Listing 6.2**). As explained earlier, the SMOTE algorithm was used to generate synthetic instances of the minority class until the number of instances in both classes were equal to each other.

In the **first phase** (*lines 10-14* in **Listing 6.2**), every file in the **STA2018** dataset (which was used to generate the models) was evaluated to select two subsets of features using the *Mean Decrease of Accuracy* and the *Mean Decrease Gini*, resulting in the formation of the MDA and

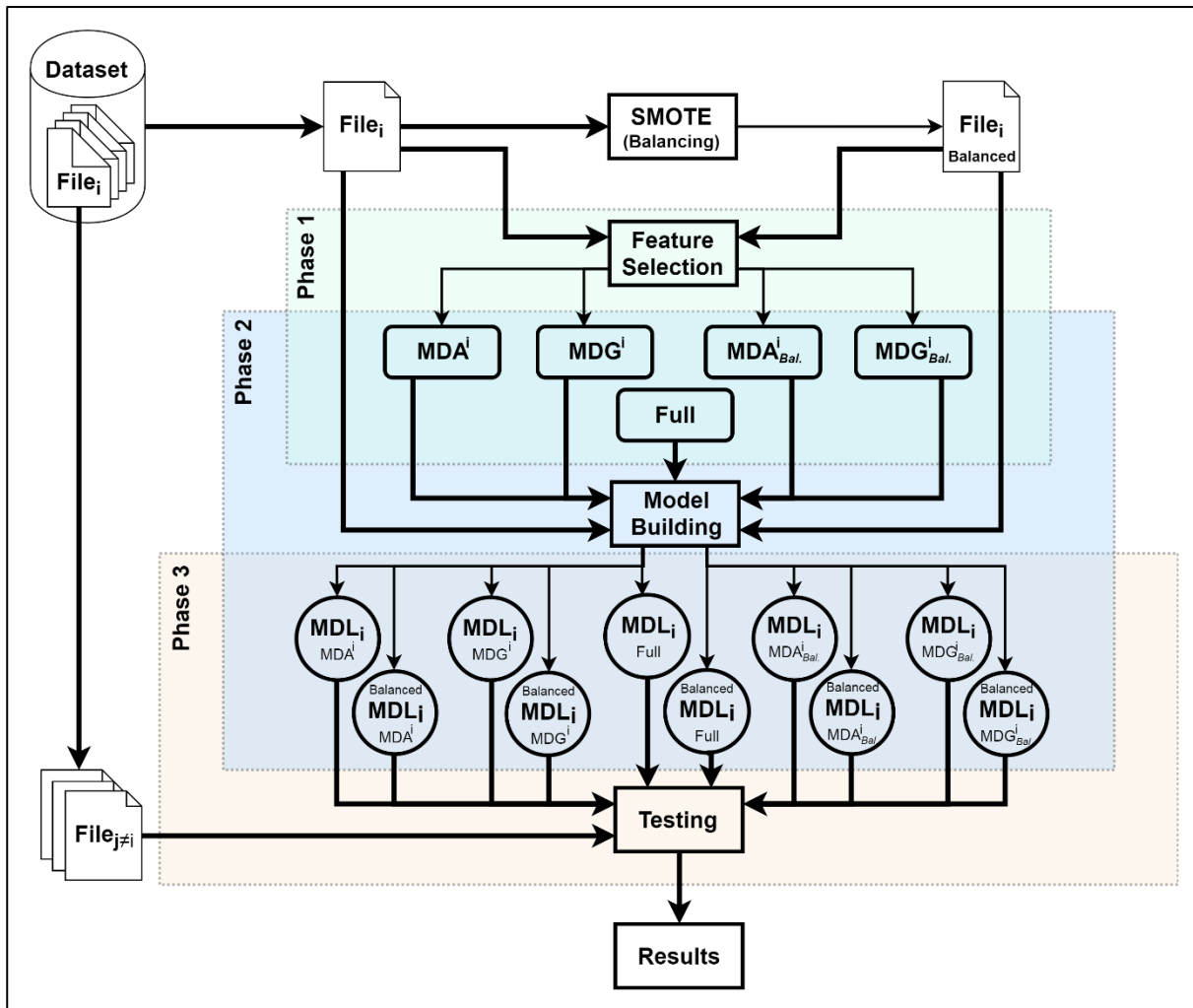


Figure 6.5: Experimental phases diagram.

MDG sets respectively. The same feature selection criteria were used on the balanced data file to generate another two sets of features, referred to in this thesis as $MDA_{Balanced}$ and $MDG_{Balanced}$. By the end of this phase there were four feature sets along with the Full features set for each training day.

In the **second phase** (*lines 16-25* in **Listing 6.2**), each day's traffic used each of the five feature sets (including the Full features set) to generate a binary classification (prediction) model which resulted in five different models. The same process was repeated using the balanced data. Each model generation step used the 3-folds Cross-Validation technique to establish the model's optimal (CV) prediction threshold. The final prediction threshold was computed by aggregating all the fold's predictions for each model to find the point (threshold) of the maximum **G-Mean**

Accuracy. By the end of this phase there were ten different binary prediction models for each day's traffic.

In the **final phase** (*lines 27-38* in **Listing 6.2**), every generated model was evaluated against each day's traffic from the dataset that had not been used in any of the feature selection or in the model generation processes. In this phase, to test the data file for each evaluation, the **G-Mean Accuracy** was computed using the model's optimal (CV) threshold and the adapted cutoff.

Algorithm: Experiment Phases

Input: Dataset

Result: Performance results

```

1  For Fi in Dataset, do           // Process every file Fi in the STA2018 dataset
2  Ftrs.Set[Full] <- {Full.Ftrs} // 544 features
3  Mdls.Set <- {}
4  Rslt.Set <- {}
5
6  Fi.bal <- Balance(Fi) // Generate/get a balanced version of data file Fi with balanced
7                          // instances' classes by generating synthetic instances of
8                          // minority class using SMOTE algorithm.
9
10 // Phase 1: features selection...
11 Ftrs.Set[MDA] <- getImportantFtrs(data=Fi, ftrType=MDA) ,
12 Ftrs.Set[MDG] <- getImportantFtrs(data=Fi, ftrType=MDG) ,
13 Ftrs.Set[MDAbal.] <- getImportantFtrs(data=Fi.bal, ftrType=MDA) ,
14 Ftrs.Set[MDGbal.] <- getImportantFtrs(data=Fi.bal, ftrType=MDG) ,
15
16 // Phase 2: models generation...
17 // Generate five predictive models using original data with five different sets of features.
18 For ftrsa in Ftrs.Set, do
19   Mdls.Set[Fi, ftrsa] <- generate.Model(data=Fi, features= ftrsa)
20 done
21
22 // Generate five predictive models using balanced data with five different sets of features.
23 For ftrsa in Ftrs.Set, do
24   Mdls.Set[Fi.bal, ftrsa] <- generate.Model(data=Fi.bal, features= ftrsa)
25 done
26
27 // Phase 3: models evaluation...
28 // Perform total of 50 evaluations (5 testing files X 10 predictive models)
29 For Fj≠Fi in Dataset, do
30   // Test every file other than Fi on every one of the 10 prediction models
31   // trained on Fi or Fi.bal
32   For Mdlb in Mdls.Set, do
33     // Get the following results:
34     // 1) G-Mean Accuracy using model's cutoff (threshold) value,
35     // 2) G-Mean Accuracy using adapted cutoff (threshold) value,
36     Rslt.Set[Fj, Mdlb] <- evaluate(data=Fj, model=Mdlb)
37   done
38 done
39
40 done

```

Listing 6.2: Pseudo code of the experimental phases.

The whole process was repeated for each of the algorithms being evaluated: C5.0, Random Forest and SVM.

Experiments were performed on a “Dell C5220 PowerEdge Rack Servers” cluster, which had 12 micro servers. Each micro server ran Scientific Linux 7 on dual quad-core Intel Xeon 3.4GHz CPUs, 16GB RAM, two 500GB SATA disks, and two Gigabit Ethernet interfaces. For the large data files [Day 5 (15/Jun) and Day 6 (16/Jun)] experiments were run on a Hyper-V virtual machine with 8 Virtual Processors, 20 GB RAM and 32 GB Swap space. This VM was used to host the Ubuntu 16.04 (64-bit) Operating System. It was hosted on a server with the following hardware specifications: 2U Supermicro chassis; 8x host-swap 2.5" SAS/SATA disk bays; Supermicro X8DTU-LN4F+ motherboard; Dual Intel Xeon E5620 (quad core); 24GB RAM (6 x 4GB DDR3 ECC RDIMM); 4x 1TB SATA (RAID10); and 4x 1Gb Ethernet. This machine uses a Windows Server 2012 R2 Datacentre (64-bit) Operating System.

6.5 Results and Discussion

The experiments outlined in this chapter started by comparing the detection performance of three well known algorithms in ML (C5.0, Random Forest and SVM) on the **STA2018** dataset with different feature sets and different data balances. Every generated model was evaluated using all of the files (subsets) in the dataset except the one that had been used to generate that model. Two G-mean accuracy (gAcc) values were computed for every combination of prediction model and evaluation data. The first G-mean accuracy (**gAcc_{Thr_{CV}}**) was the one obtained after the model’s optimal (CV) cutoff value had been calculated using 3-folds Cross-Validation to predict the data file. The other G-mean accuracy value (**gAcc_{Thr_{Opt}}**) was calculated based on the maximum accuracy achieved after the prediction cutoff value had been specifically adapted for the evaluated data file.

This section starts by looking at the effect of the cutoff adaptation by determining the statistical significance in the performance (G-Mean accuracy) of the models through comparing their optimal threshold with the adaptive cutoff. The analysis compared the difference between the two approaches by conducting four Friedman's tests (with a significance level of $\alpha = 0.05$): the *first* compared the overall difference between the two approaches; the *second* compared the difference between the two approaches using different algorithms; the *third* compared the difference between the two approaches using different feature sets; and the *fourth* compared the difference between the two approaches using different data balances.

The following list shows the hypotheses that were tested and the results returned by the Friedman tests. [The decision to use the non-parametric Friedman's test was based on the fact that the data did not follow a normal distribution, as confirmed by the normality test (Shapiro–Wilk test) [332] $W = 0.7$, $p\text{-value} = 0.000$.]

Threshold- H_0 : “*there are no statistically significant differences in model performance (G-Mean accuracies) before and after cutoff (threshold) adaptation has been applied.*”
 $\chi^2(1) = 873.0$, $p = 0.000 < 0.05$ (*differences are statistically significant*)

ML- H_0 : “*there are no statistically significant differences in model performance (G-Mean accuracies) between the different ML algorithms (C5.0, RF and SVM) before and after cutoff (threshold) adaptation has been applied.*”
 $\chi^2(5) = 747.5$, $p = 0.000 < 0.05$ (*differences are statistically significant*)

Features- H_0 : “*there are no statistically significant differences in model performance (G-Mean accuracies) between the different feature sets (Full, MDA, MDG, MDA_{Bal.} and MDG_{Bal.}) before and after cutoff (threshold) adaptation has been applied.*”
 $\chi^2(9) = 742.8$, $p = 0.000 < 0.05$ (*differences are statistically significant*)

Balance- H_0 : “*there are no statistically significant differences in model performance (G-Mean accuracies) between the different data balances (Original and Balanced data) before and after cutoff (threshold) adaptation has been applied.*”
 $\chi^2(3) = 761.3$, $p = 0.000 < 0.05$ (*differences are statistically significant*)

As all of these tests showed significant differences, a Nemenyi post-hoc test [373-375] was conducted to perform pairwise comparisons on the different effects of each test to distinguish which differences were statistically significant. The results of these pairwise comparisons are illustrated in **Figure 6.6** through critical difference plots.

All of the plots in **Figure 6.6** show that the cutoff adaptation effect was significantly different from the model's optimal (CV) threshold. They also show that different treatments (*ML algorithm, feature sets and/or data balance*) with the adaptive cutoff always ranked higher. Any insignificant differences fall within the same effect (cutoff adaptation or model's optimal threshold). For example, **Figure 6.6-(b)** shows that there were insignificant differences between SVM and C5.0 when the cutoff adaptation was applied as well as between RF and SVM when the model's optimal threshold was used. Overall, no two treatments of different groups (optimal or adaptive cutoff) showed any insignificant differences.

Having shown that the models' performance was ranked significantly higher when the adaptive cutoff approach was used rather than the optimal (CV) threshold [See **Table B.1**, **Table B.2**, **Table B.3**, **Figure B.1**, **Figure B.2** and **Figure B.3** in **Appendix (B)**], all subsequent analyses focus on the results obtained from using the adaptive cutoff.

Each plot in **Figure B.1**, **Figure B.2** and **Figure B.3** [presented in **Appendix (B)**] shows the performance (G-Mean Accuracy) for each model for each training day for the C5.0, RF and SVM algorithms respectively. These plots show each model's performance under different feature sets (Full, MDA, MDG, MDA_{Bal.} and MDG_{Bal.}) and data balances (Original and Balanced). Each plot is composed of 10 sub-plots (one for each model) and illustrates the G-Mean Accuracy for each model after being evaluated using all the other days' files. For each evaluation there are two G-Mean Accuracy readings; one is based on the model's optimal threshold while the other uses the adapted threshold on the test data. They are represented by 'CV Cutoff' and 'Adp. Cutoff' curves respectively. In these sub-plots, the first day (along the x-axis) matches the training day of the main plot and corresponds to the CV results of that model. In general, all these plots show very clearly that the cutoff adaptation process improved the performance of the models, while the model's optimal threshold led to low levels of accuracy compared to the capabilities of the true model.

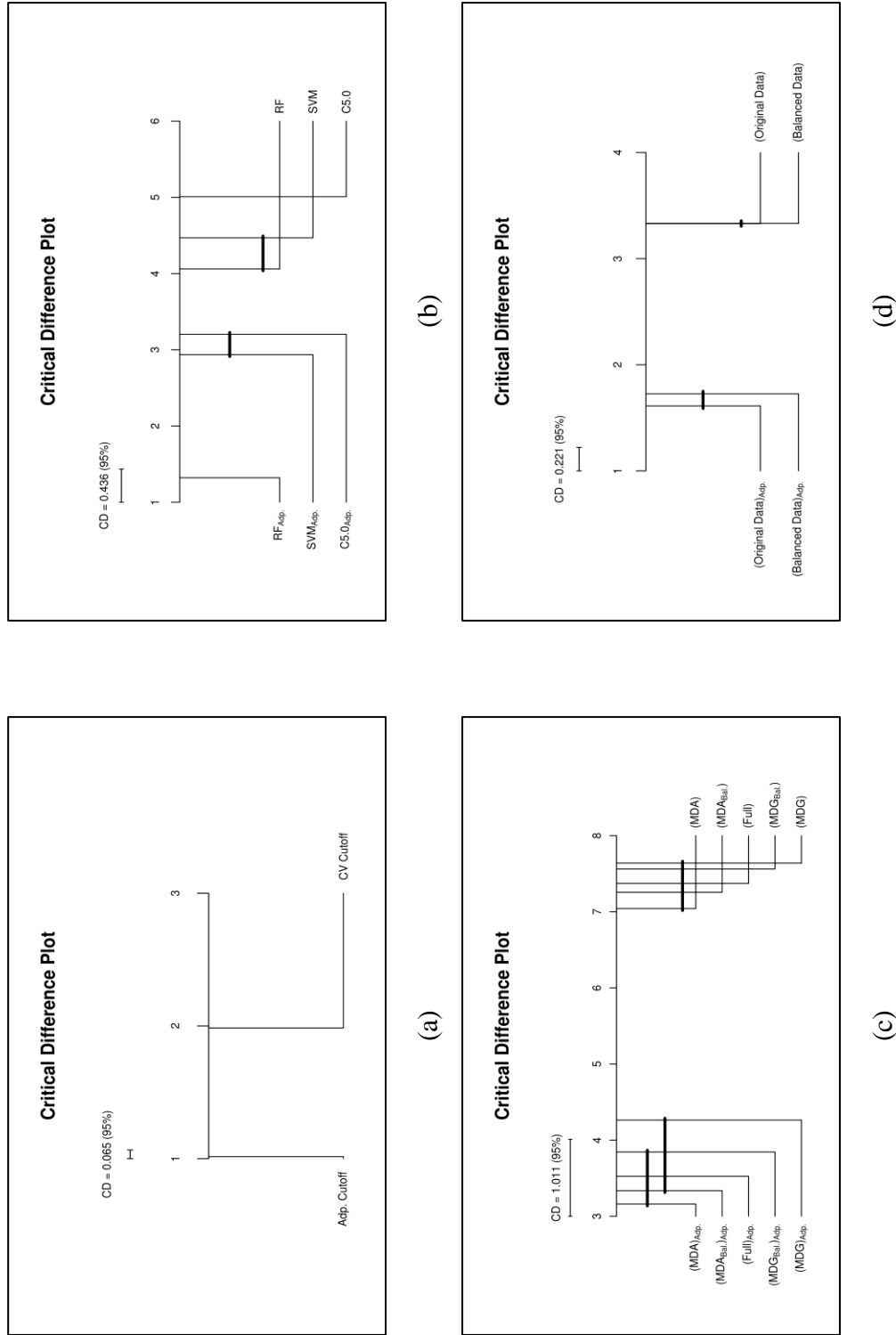


Figure 6.6: Graphical illustration of pairwise comparisons from the Friedman Test results for different threshold effects (optimal or adaptive cutoff) after applying the Nemenyi test (95% confidence level) (a) Nemenyi test for different thresholds. (b) Nemenyi test for different feature sets. (c) Nemenyi test for different algorithms. (d) Nemenyi test for different training data balances.

6.5.1 C5.0 Algorithm

The plots in **Figure B.1** [see **Appendix (B)**] for the C5.0 models show different patterns and behaviours from one training day to another. For example, models trained on Day 2 (12/Jun) failed to perform well on Day 5 (15/Jun), whereas Day 5 models predicted Day 2 traffic with a high degree of accuracy. They also showed inconsistent behaviour towards different feature sets across the days. For example, Day 2 models performed best when the Full feature set was used. But this pattern was not consistent across all days. This can clearly be seen on the plot of Day 5 when MDG features were used, and the plot of Day 7 (17/Jun) when MDA or MDA_{Bal.} feature sets were used with the balanced training data. One important observation to make is the poor performance of Day 6 (16/Jun) models when the original training data were used. These models showed the worst performance due to the low number of attacks in this data file. When a balanced version of the Day 6 data file was used to build the prediction models, performances improved. This supports the finding discussed in **Chapter 4** about the behaviour of C5.0 algorithm with imbalanced data. It can also be clearly observed from these plots that data balancing had a minor effect in improving the performance of models developed using the C5.0 algorithm, which was further investigated using statistical analysis.

The adapted threshold provided a more accurate reading of a model's true performance. **Table 6.6** summarises the performance of all the models [which are set out in full in **Table B.1** in **Appendix (B)**]. It presents the average performance for each model on the evaluation data using the adapted (cutoff) threshold. In the table each row related to the C5.0 algorithm shows the average performance for every model presented in **Table B.1** using the adapted cutoff on the evaluation data, omitting any performance obtained at the CV stage. The values in **bold** are the maximum value for each group of the training data (original or balanced) while the value that is underlined is the maximum value for that day. For example, the average of the G-Mean

		Original Data					Balanced Data				
		Full	MDA	MDG	MDA _{Bal.}	MDG _{Bal.}	Full	MDA	MDG	MDA _{Bal.}	MDG _{Bal.}
C5.0	Day 2	0.6202	0.2159	0.2159	0.2159	0.2159	0.7091	0.2159	0.2159	0.2159	0.2159
	Day 3	0.6904	0.7464	0.7428	0.8693	0.5681	0.7351	0.8660	0.8190	0.7587	0.8124
	Day 4	0.5941	0.5837	0.9750	0.7104	0.8424	0.7854	0.7401	0.7311	0.7434	0.7703
	Day 5	0.7359	0.6089	0.7994	0.7582	0.5901	0.7494	0.7666	0.9378	0.7623	0.7153
	Day 6	0.0000	0.0000	0.0000	0.0000	0.0000	0.4442	0.7932	0.6600	0.7925	0.6600
	Day 7	0.7423	0.8719	0.8698	0.8707	0.8762	0.7227	0.9444	0.9092	0.9444	0.8856
	RF	Day 2	0.9566	0.9606	0.9404	0.9698	0.9618	0.9590	0.9353	0.9632	0.9665
Day 3		0.9819	0.9786	0.8616	0.9815	0.9241	0.9747	0.9755	0.8065	0.9711	0.9436
Day 4		0.9880	0.9875	0.9014	0.9872	0.9693	0.9831	0.9844	0.8908	0.9828	0.9716
Day 5		0.9715	0.9700	0.9524	0.9709	0.9537	0.9671	0.9708	0.9348	0.9674	0.9658
Day 6		0.8905	0.7305	0.7874	0.8491	0.8540	0.7313	0.7201	0.7385	0.7372	0.7066
Day 7		0.9716	0.9723	0.9719	0.9720	0.9728	0.9691	0.9666	0.9690	0.9675	0.9729
SVM		Day 2	0.5441	0.7716	0.7712	0.8036	0.8820	0.8747	0.8653	0.7275	0.8375
	Day 3	0.8149	0.5564	0.4976	0.5615	0.8143	0.7624	0.9035	0.5528	0.9032	0.6618
	Day 4	0.7971	0.8472	0.9500	0.8294	0.8703	0.8927	0.8981	0.9654	0.8902	0.8508
	Day 5	0.7292	0.6807	0.6436	0.6666	0.6841	0.6516	0.6264	0.5246	0.6699	0.5846
	Day 6	0.8069	0.9558	0.8595	0.8062	0.8049	0.6390	0.8646	0.8588	0.8614	0.8688
	Day 7	0.7097	0.7284	0.8540	0.6720	0.8548	0.7363	0.7638	0.9032	0.7640	0.7847

Table 6.6: Model's average performance for different ML algorithms, feature sets and data balances.

Accuracy values for the Day 2 model, using the *Full* features set, trained on the *original data* is 0.6202 (which is the maximum value for all of the models within the original data group for that day). Applying the same process to the balanced data group revealed that, for that group, the *Full* features set had a maximum average of 0.709. As this value is the maximum for all models for that day it has been underlined as well.

Friedman's test was used to assess whether the performance of the C5.0 algorithm's models using these features sets with different data balances would be significant after a threshold adaptation with a significance level of $\alpha=0.05$ was applied. The tested hypothesis was, "*there are no statistically significant differences in the performance (G-Mean accuracies) of models built using the C5.0 algorithm with different feature sets and different data balances after a cutoff (threshold) adaptation has been applied.*" This test revealed that there was not enough evidence to support this hypothesis, $\chi^2(9) = 16.0$, $p = 0.067 \not\leq 0.05$.

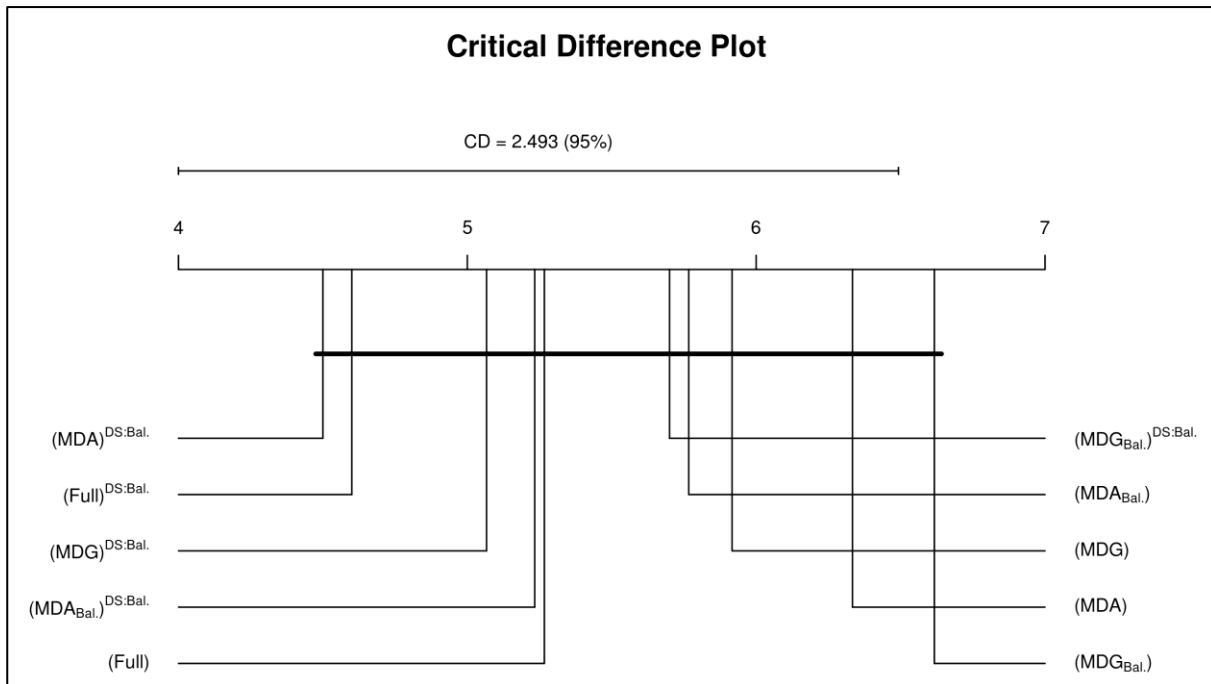


Figure 6.7: Nemenyi test (95% confidence level) on the C5.0 algorithm models using different feature sets and different data balances after applying the adaptive cutoff approach.

These tests show that there was no significant effect of one feature set over another when the C5.0 algorithm was used. In addition, data balancing did not lead to a significant improvement in a model's performance. **Figure 6.7** shows that the performance of models built using the balanced training data ranked higher than those generated using the original data. However, these differences were not statistically significant at the 95% confidence level, as all of the effects are joined by a line. **Figure B.1** in **Appendix (B)** shows that different behaviours were exhibited on different days. Models trained on some days performed well when the data was balanced, whereas other days showed no sign of improvement. Some days appeared to have been affected by one features set, while other days behaved the same for all feature sets.

Many factors could be behind the volatile behaviour of the C5.0 algorithm. For example, selected feature sets might not be the best sets for this algorithm; as such, further investigation into different feature selection techniques is required. Also, this algorithm was executed within its default parameters, particularly number of trials, which was set at **ten**. Further tuning of the number of trials would also benefit from further analysis. In addition, C5.0 algorithms carry out random sampling by following the boosting technique (which randomly samples weighted

instances). This might have caused C5.0 to overfit the training data which could be one of the reasons for its overall poor performance in predicting new traffic. Overall, based on the statistical results returned using Friedman’s test, the C5.0 models ranked low, as illustrated by **Figure 6.6-(b)**.

6.5.2 RF Algorithm

Another Friedman test was performed to assess the RF algorithm’s models. This test aimed to determine how these models performed when using different feature sets with different data balances, and whether the difference in performance was significant after applying the threshold adaptation with a significance level of $\alpha=0.05$. The tested hypothesis was, “*there are no statistically significant differences in the performance (G-Mean accuracies) of models built using the RF algorithm with different features sets and different data balances after the cutoff (threshold) adaptation has been applied.*”

This test revealed that for the **RF** algorithm there were significant differences between these features after applying the *cutoff (threshold) adaptation*, $\chi^2(9) = 38.0$, $p = 0.000 < 0.05$. To distinguish which of these effects were statistically significant a Nemenyi post-hoc test was conducted to perform a pairwise comparison as illustrated in **Figure 6.8**.

Overall, there were no significant differences in the RF’s performance when Full, MDA and MDA_{Bal.} feature sets were used. However, the Full features set showed a significant difference over the MDG and MDG_{Bal.} feature sets, which ranked lowest among the feature sets. This could be due to the nature of the *Mean Decrease Gini* in selecting local features which have low generalisation power as discussed earlier. However, even with these low performances, RF had the highest overall performance. As **Figure 6.8** shows the data balance had no significant effect on the performance of RF. On the contrary, it sometimes negatively affected the performance of models using the Full feature set with balanced data as their difference to the

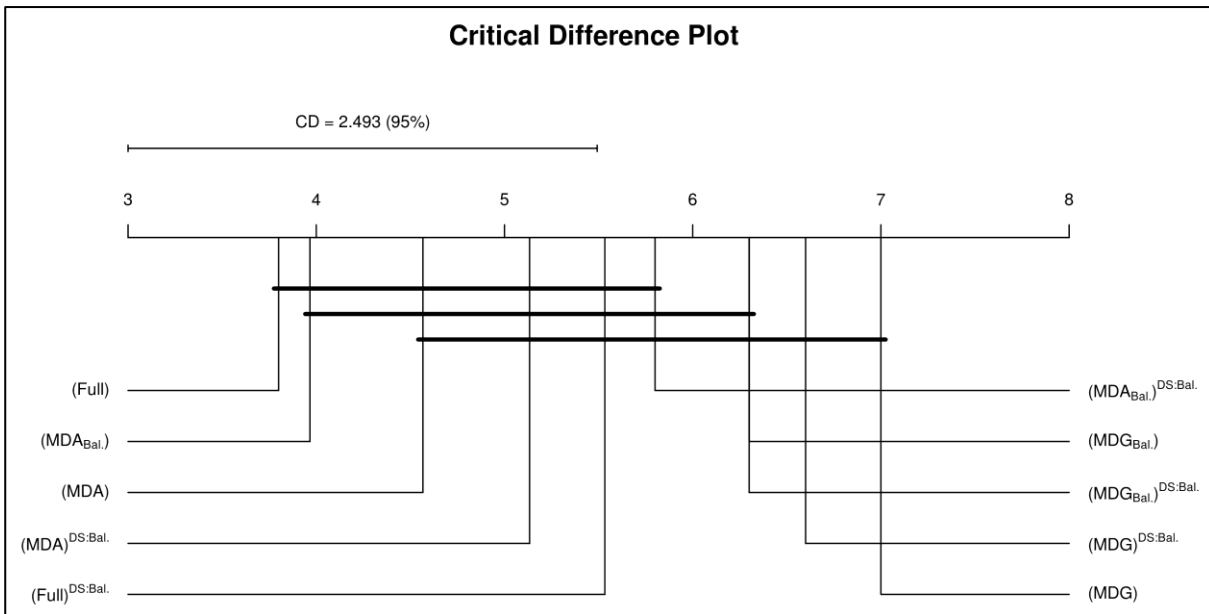


Figure 6.8: Nemenyi test (95% confidence level) on the **RF** algorithm models using different **feature sets** and different **data balances** after applying the adaptive cutoff approach.

MDG and MDG_{Bal.} feature sets became insignificant. This is also evident in the plot for Day 6 in **Figure B.2** [set out in **Appendix (B)**] which showed a lower performance for all models for that day as the balanced version of data was used. Although that day only had 11 attacks, RF was able to build good predictive models with good evaluation performance except for Day 4's traffic. The ability of RF to learn from Day 6 traffic was linked to its bagging technique, which randomly samples instances that are used to build its trees. In contrast to C5.0, this sampling technique prevented RF from overfitting, which in turn produced models with good generalisation capabilities. This gave RF more chance of detecting novel attacks as demonstrated in these experiments.

The Random Forest (RF) algorithm showed the best results of the evaluated ML algorithms. As illustrated in the plots in **Figure B.2** [in **Appendix (B)**], RF's performance would not have been better than that of the other algorithms, if the optimal (CV) threshold of its models had been used to assess their performance. However, with the cutoff adaptation approach, RF's performance improved significantly as **Table B.2** [in **Appendix (B)**] shows.

The RF algorithm can take longer to train depending on the complexity of the training data. However, once the model is built, its evaluation of a new instance is reasonably fast.

As expected, it consumed a lot of resource (memory) at the model building phase and this consumption increased with the size of the training data. This was a result of the number of bootstrap samples it generated, which were used to build trees in parallel threads. The resulting models were quite large compared to the SVM and C5.0 models, and their sizes increased as the complexity of the training data increased.

Although **Table 6.6** shows the highest performance of the RF models was attained when the Full features set was used, the difference in the average performance of its models was very small, unlike the performance of the C5.0 and SVM models, which showed higher variations in performance. Therefore, RF models could be generated using a reduced feature set without any significant decrease in their average performance (accuracy) but with a significant gain in speed. **Table 6.6** also shows that there was only a high variation in the performance of models for Day 6; however, given that this day was problematic, with its skewed balance, this level of performance is more than acceptable. Moreover, in a real life scenario it would not be sensible to build a model using such data, hence this example is an extreme case, which is presented here merely to demonstrate that the RF algorithm performed reasonably well.

6.5.3 SVM Algorithm

To assess the difference in the performance of the SVM model using the different feature sets with different data balances after applying the threshold adaptation, the Friedman test was used with a significance level of $\alpha=0.05$. The tested hypothesis was, “*there are no statistically significant differences in the performance (G-Mean accuracies) of models built using the SVM algorithm with different feature sets and different data balances after the cutoff (threshold) adaptation has been applied.*” This test revealed that there was **not** enough evidence to support this hypothesis, $\chi^2(9) = 13.1, p = 0.158 \not\ll 0.05$.

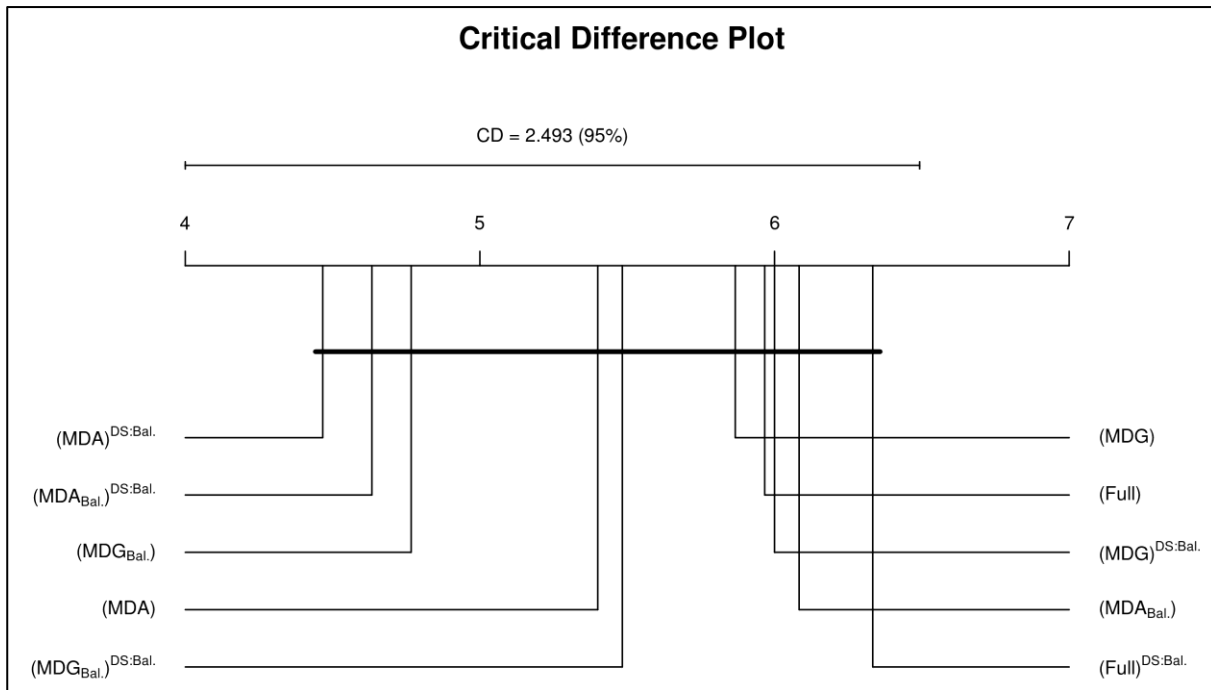


Figure 6.9: Nemenyi test (95% confidence level) on SVM algorithm models using different feature sets and different data balances after applying the adaptive cutoff approach.

The SVM algorithm exhibited similar behaviour to the C5.0 algorithm. All of its statistical tests revealed insignificant effects between one feature set and another and there was no sign that the improved performance of its models was influenced by any of the data balancing effects. Although **Figure 6.9** shows that the reduced feature sets ranked higher for this algorithm than for the Full features set, these differences were statistically insignificant. As with the C5.0 algorithm, different behaviours were exhibited on different days, as illustrated in **Figure B.3** [in **Appendix (B)**], so no consistent pattern could be deduced.

Although the SVM algorithm showed some overall improvement on days when the reduced feature sets were used instead of the Full features set, this behaviour was not consistent. As a linear version of SVM was used, this effect could have been caused by the non-linear nature of the data on those days where SVM failed to perform well. Further investigation would be required to analyse different kernel transformations of the data to determine the best tuning parameters or implementation for this algorithm which would fit this domain. Such an investigation did not take place as part of this study because of the limitations of SVM for non-linear problems. As kernel functions were used, SVM would take longer to build its prediction

models and longer to predict new instances. This problem might be an obstacle to introducing SVM into a dynamic and high volume environment, such as network analysis and ID.

In general, although a linear SVM implementation was used in these experiments, it showed some good results. For example, on average, the performance of models trained on the original and balanced version of Day 4's traffic, using MDG features, was above 90%. These results are presented in **Table 6.6** which sets out the average performance of the SVM models. Also the performance of models trained using the MDA features on the original version of Day 6 traffic (which only had 11 attacks), was above 89%. With such results, more analysis would be required to identify the right combination of fast kernel function and parameter tuning to improve the overall SVM results. This would make it an attractive solution for IDS problems.

Figure 6.10 summarises all of the figures in **Appendix (B)** (**Figure B.1**, **Figure B.2** and **Figure B.3**) and all of the accuracy readings in the tables (**Table B.1**, **Table B.2** and **Table B.3**) after the threshold adaptation process was applied. It compares the average performance of all the C5.0, RF and SVM models. This plot shows the average performance for each day's model for all of the tested ML algorithms. The standard error of the average performance for each model is illustrated by vertical bars. For each algorithm, the mean performance of all models across all days for every combination of feature sets and data balance type is represented by a horizontal dashed line. As this plot shows, RF was always the highest performing of the ML algorithms evaluated. Unlike C5.0 and SVM, RF showed the most stable results with the least variability. A similar conclusion was reached by Japkowicz and Stephen [221] regarding the sensitivity of the C5.0 algorithm (see **Section 2.2.3.2**).

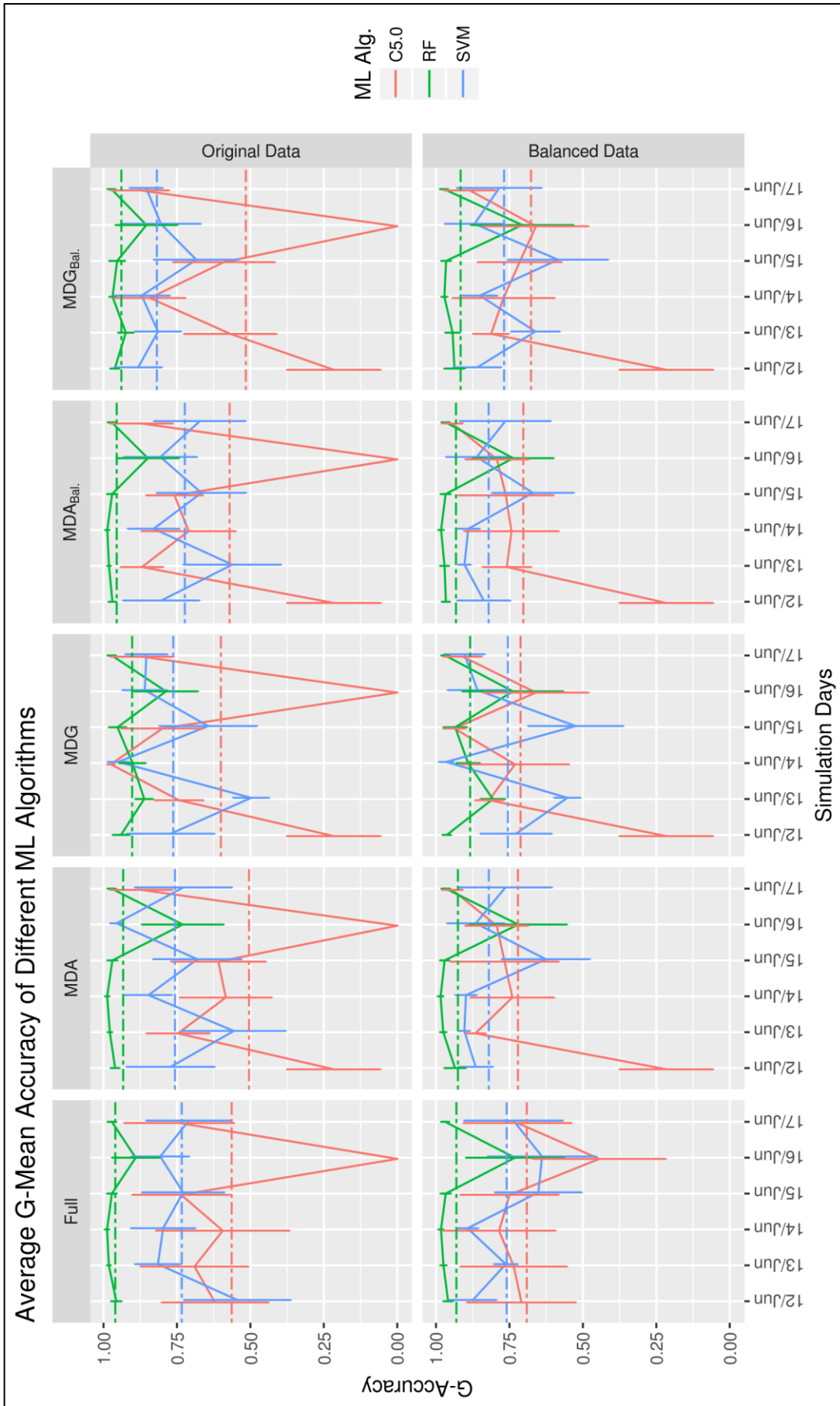


Figure 6.10: Comparison plot of the average performance of every C5.0, RF and SVM model for every feature set and data balance combination.

6.6 Limitations

Although the findings of these experiments support the findings discussed in **Chapter 4**, i.e. that applying a cutoff adaptation to the evaluation (testing) data is important for achieving an accurate reading of the true performance of a prediction model, there were a number of unavoidable limitations to these experiments.

Firstly, some limitations are similar to those discussed in **Chapter 4**, such as the sole focus on binary classification problems. Also, the best setup for the parameters of different ML algorithms requires further analysis, such as the optimal number of trials for the C5.0 algorithm and the use of other non-linear implementations of SVM. These tuning requirements could be considered engineering issues which require further research in their own right.

Secondly, time constraints played an important role in the decisions taken in relation to the experimental settings. For example, a decision was taken to use 3-folds Cross-Validation instead of 10-folds Cross-Validation because of the gain in execution time. Further investigation into the optimal number of folds could be conducted in a separate study to determine the best balance between time and accuracy for production environment.

Thirdly, only two feature importance measures were used in the feature selection stage: the *Mean Decrease of Accuracy* and the *Mean Decrease Gini*. Both of these measures were computed using the RF algorithm, which could have made the selected features ideal for this algorithm but not for the others. As the results show, with these measures the RF algorithm's performance was nearly the same across all selected features. Although this could be regarded as a limitation, it could equally be argued that each day file had different important features and despite these differences, RF was able to generalise even when the evaluation data had different important features. Further investigation is required into what other feature selection techniques could be used, especially on algorithms other than those evaluated in this study. There are many

techniques that could be explored, such as correlation-based feature subset selection²⁰ [200], the use of information gain as an evaluation criterion for the features' importance²¹ and the minimum-redundancy-maximum-relevance (mRMR) feature selection [190, 376, 377].

In addition, there was no analysis of the relationship between the selected features and the performance of their models. For example, an analysis of the effect of features common to different days could have identified which features were tightly linked to the performance of different models. This line of research was not followed as it would have required an analysis of the full dataset, and the main aim of this study was to mimic real life scenarios where future traffic is unknown and decisions need to be made on the (training) data available.

Fourthly, only one balancing technique was adopted for these experiments. An evaluation of different data balancing techniques could determine which technique would best fit the network ID domain. Using the SMOTE algorithm to generate synthetic connections might not be an ideal solution for this domain as the connections generated might not represent a valid real connection. Further analysis is required to determine whether there is any relation between the balancing technique chosen and the performance of the ML algorithm.

Fifth, the analysis outlined in this chapter was limited because of the structure of the UNB ISCX2012 dataset. A day was used as the window size to split the traffic into subsets for training and evaluation purposes. To allow a deeper analysis of the effect of the time window size (i.e. to extend the time window to weeks or months) on the performance of this approach, a larger dataset would be needed. Although reducing the time window to hours would have been a valid option, this approach was not taken because the formation of pure traffic within small window sizes would have shifted the focus to another domain of research, such as data stream classification. Also, as this analysis was limited to one dataset due to time constraints, further

²⁰ <http://weka.sourceforge.net/doc.dev/weka/attributeSelection/CfsSubsetEval.html>

²¹ <http://weka.sourceforge.net/doc.dev/weka/attributeSelection/InfoGainAttributeEval.html>

investigation would be required to analyse the effectiveness of this approach on datasets with diverse traffic. The decision to use this dataset was based on the limited number of existing transformed datasets, given that the transformation and labelling of a new reliable dataset is a time consuming task.

Finally, the feature selection process used a statistical comparison to compare the mean importance of every evaluated feature with the mean performance of the fake variables. This evaluation assumed that the importance measures would follow a normal distribution which led to the use of Welch's two sample t-test. Further analysis is required to evaluate a dynamic approach of applying the appropriate test (parametric or non-parametric) based on a data normality check.

6.7 Summary

This chapter has presented a set of experiments undertaken to analyse the effect of applying cutoff adaptation to evaluation data on the performance of three main ML algorithms: C5.0, RF and SVM. The analysis investigated the effect of feature selection and data balancing on the overall performance of models developed using these algorithms before and after cutoff adaptation. These experiments aimed to simulate real-life setups in terms of how they conducted their model building and evaluation.

This analysis built models on subsets (traffic for one day) of the data to predict the remaining parts (traffic for other days). The results of these analyses showed that RF outperformed the other algorithms in its ability to predict new traffic and the detection of novel anomalies. It also showed that before cutoff adaptation, all of the ML algorithms performed as poorly as each other, but that the adaptive cutoff approach increased their overall performance, with RF performing best. Moreover, RF suffered no significant loss in performance when the reduced feature sets were used and its predictions did not improve when the data was balanced. This

gives RF the advantage of being able to build models using original data with a reduced feature set, which will save a considerable amount of time in training and testing, which makes this algorithm more attractive for such problems.

In these analyses, G-Mean accuracy measures were used as the model assessment criteria to avoid issues with imbalanced data. The performance of all models was assessed using the non-parametric Friedman test to identify any significant differences. Cutoff adaptation and the algorithm used were the most important effects that contributed to any significant difference in a model's performance.

Having established the importance of cutoff adaption in determining the performance of a prediction model, the next chapter will look at a technique to determine the appropriate cutoff for any dataset being evaluated based on a small randomly selected subset. This subset will then be labelled and used to set the right adapted cutoff for the whole dataset, even when, as in a real life problem, all its true labels are not known.

Chapter 7

Cutoff Selection Based on Evaluating a Subset of the Test Data

This chapter investigates the selection of an adaptive cutoff (threshold) for evaluation (test) data, based on the true labels of a random subset, i.e. validation data. It also includes an analysis of the effect of different **sampling sizes** in determining the right threshold (cutoff) for the predictions of different ML models. The analysis also evaluated different **sampling approaches** to assess their ability to identify the correct threshold (cutoff) for the whole dataset.

This analysis was conducted using the results of the experiments that were outlined in the previous chapter (**Chapter 6**). The G-Mean Accuracy Ratio, which measures the ratio of the performance (G-Mean Accuracy) of the sample cutoff relative to that of the optimal cutoff of the full test data, was used as the metric to compare the different effects. Further investigations were carried out to determine how different **error rates** (introduced to the labels of the random small subsets) might affect the identification of the correct discriminating threshold (cutoff).

7.1 Introduction

Previous chapters have demonstrated the importance of the threshold (cutoff) adaptation to achieve an accurate reading of a model's performance and to improve predictions in real environments. As the example illustrated in **Figure 7.1** shows, the disadvantage of using a fixed threshold, such as $\text{Thr}_{0.5}$, is that it undermines the capacity of some good prediction models which could result in the selection of weaker models. **Figure 7.1** presents an example of the

performance (measured by G-Mean Accuracy) of three dummy predictions (Pred₁, Pred₂ and Pred₃). It shows that Pred₁ delivers higher levels of prediction accuracy at low threshold values, but that using a fixed threshold, i.e. Thr_{0.5}, subverts its true capacity. Pred₂ performs the least well out of all of the predictions, however, that fixed threshold gives a false reading of its true performance and shows a marked difference in its levels of accuracy compared to Pred₁. Independently adapting the discriminating threshold for every prediction result leads to more accurate readings of a model's true potential (performance). Some predictions such as Pred₃ do not show any significant difference between a fixed and an adapted threshold, however, the adapted threshold usually offers a higher degree of accuracy. This is not just the case for fixed thresholds such as Thr_{0.5}, as other fixed thresholds behave the same way. As has been shown in **Chapter 6**, threshold adaptation improved the overall performance of all of the prediction models and showed their true potential.

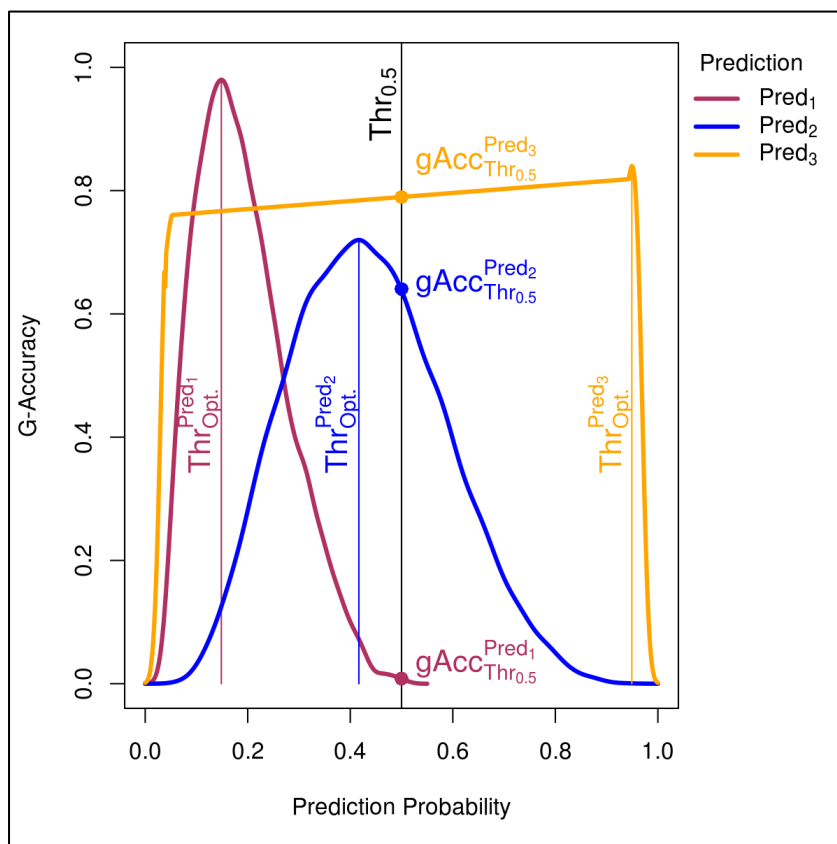


Figure 7.1: Example comparing fixed threshold (solid black vertical line - Thr_{0.5}) with adapted prediction thresholds (maroon, blue and orange vertical lines).

The threshold (cutoff) adaptation process outlined above is based on the ground truth (true labels) of the evaluation data. Unfortunately, this requirement is not available in real life situations. Therefore, this chapter investigates the potential of determining the best prediction threshold (cutoff) for an evaluation data using the true labels of a small random sample only.

In the ID domain, the labelling of the small subset can be viewed as an evaluation of a sample of connections to determine their true states (labels) using the assessment of a security specialist or the classification of a signature based IDS. This subset will be used to set the correct threshold (cutoff) for all the traffic, which will then be used to flag anomalies. Hence, these anomalies can be investigated and analysed to identify novel and unknown attacks, which could be used to update the signature-based IDS.

Random sampling from the population does not ensure that the sampled data covers the full range of the prediction probabilities in the population; this is especially the case when a model produces predictions with low probabilities for unseen data and when the threshold needs to be adapted for such new test data. Therefore, various sampling strategies, using a binning technique, were used for the experiments outlined in this chapter. To address the original probability distribution of the predictions range of a model on the test data, the range was divided into 'bins'. Random sampling from every bin thus ensured that the prediction probability distribution for each sample was close to the probability distribution of the whole test data. The experiments outlined below compare the performance of different bin sizes to a normal random sampling approach from the whole population.

These experiments aimed to simulate a real life situation where the labelling of subset samples would be undertaken by external sources, such as a security expert or a signature-based IDS. These labels were then used to assess the best cutoff for the whole of the test data. This labelling process is prone to errors due to a number of external factors, such as human error or false alarms by the IDS. As a result, these experiments attempted to analyse the effect of different

error levels on the process of identifying the best threshold (cutoff). This chapter also looks at the effect of different sample sizes in identifying the best cutoff, of the entire test data, based on the sampled subset. One of the aims of the experiments was to show the trade-off between sample size and the correct setting of the cutoff value. This is because the traffic load in a real life network is enormous, so finding the best cutoff, based on the smallest sample size, would save a great deal of effort and resource.

7.2 Proposed Solution

This chapter looks at the selection of an adaptive cutoff (threshold) for evaluation data where the true label is only known for a small, random sample. It provides an analysis of the effect of different sample sizes (10%, 5%, 1%, 0.5%, 0.1%, 0.05%, 0.01%, 0.005%, 0.001%, 0.0005% and 0.0001%) on the selection of the right threshold (cutoff). It is worth pointing out that the percentage values should be controlled to draw enough samples that would be sufficiently representative of the original data (order of magnitude). As the analysed datasets in this study are large, the use of these small percentages was sufficient. However, for smaller datasets, larger percentages might be needed.

It is important to note that these experiments stress the importance of performing the sampling of instances based on the predictions of models rather than the sampling from the original connections population. This sampling approach is required to capture the distribution of the predictions which would not have been fully captured when sampling is performed randomly from the population. In other words, random sampling, from the entire test data, could miss the small number of anomalous cases as it does not take the full range of the model's predictions into account. As a result, a binning sampling strategy is applied and compared with the normal random sampling approach. In the binning approach, the range of a model's predictions is divided into B bins (1, 10, 20, 50 and 100). Samples were then randomly selected from each

bin to ensure good coverage of rare cases. The **one-bin** (B_1) type is the random sampling from the whole population.

To address the possibility of errors in labelling and in determining the state of sampled connections (due to human errors or the inability of signature-based IDS to detect novel attacks) different error rates (0%, 1%, 5% and 10%) were introduced to the true labels of the random samples. This aimed to identify the effect of such errors -in sample labels- in determining the best prediction threshold (cutoff) and what is expected on the overall performance of the adaptive cutoff approach.

These experiments were evaluated using a G-Mean Accuracy Ratio (GAR), which assesses the quality of the sample cutoff (Thr_{Smp1}) by measuring its closeness to the optimal threshold (Thr_{Opt}). The GAR measure computes the ratio of the classification performance (G-Mean Accuracy) of the data, using the sample cutoff (Thr_{Smp1}), to that of the optimal cutoff (Thr_{Opt}), as presented in **Eq.(7.1)**.

$$\text{G-Mean Accuracy Ratio (GAR)} = \frac{\text{g. accuracy(Data | } Thr_{Smp1})}{\text{g. accuracy(Data | } Thr_{Opt})} \quad \text{Eq.(7.1)}$$

The GAR measure ranges between zero and one; the closer the sample cutoff (Thr_{Smp1}) to the optimal threshold (Thr_{Opt}), the closer the GAR will be to one. As Thr_{Smp1} shifts away from Thr_{Opt} the GAR will be closer to zero. **Figure 7.2** illustrates this with an example of the G-Mean Accuracy Ratios (GAR) of two dummy predictions. This example shows that the GAR for $Pred_2$ (0.94) is higher than it is for $Pred_1$ (0.90), even though the optimal performance of $Pred_1$ is higher than that of $Pred_2$. This is because this measure assesses how good the selected cutoff is in relation to the optimal cutoff for a prediction result, not for the overall prediction performance.

The effect of different sampling strategy under every tested error rates on selecting the ‘close to optimal’ threshold has been analysed. This is done by evaluating the following hypothesis

on the predictions of ID models for every ML algorithm: “*there are no statistically significant differences in the G-Mean Accuracy Ratios (GAR) of the sample cutoffs of the {C5.0, RF, SVM} model predictions between the different sampling strategies –number of bins- (B_1 , B_{10} , B_{20} , B_{50} and B_{100}) when the sample labels have an {0%, 1%, 5%, 10%} error rate”.*

Similar analysis is performed to evaluate the effect of different sample sizes, by testing the following hypothesis on the predictions of ID models for every ML algorithm: “*there are no statistically significant differences in the G-Mean Accuracy Ratios (GAR) of the sample cutoffs of {C5.0, RF, SVM} model predictions between the different sample sizes (10% to 0.0001%) when the sample labels have an {0%, 1%, 5%, 10%} error rate”.*

Finally, the effects of different error rates on the selection of the ‘close to optimal’ threshold on the predictions of the ID models of every ML algorithms is analysed by testing the following hypothesis, “*there are no statistically significant differences in the G-Mean Accuracy Ratios (GAR) of the sample cutoffs of the different ML (C5.0, RF and SVM) model predictions at different error rates (0%, 1%, 5% and 10%) in the sample labels”.*

7.3 Experimental Setting

The experiments presented in this chapter used the prediction results of the models developed in **Chapter 6** on the **STA2018** dataset. As noted earlier, the main aim of these experiments was to evaluate the effectiveness of the threshold (cutoff) setting using a randomly selected subset of the test data. The true label of this subset was then used to compute the best cutoff value (Thr_{smp1}) at which the maximum G-Mean Accuracy could be achieved on this sample. The Thr_{smp1} was used to classify the original test data and to compute its classification performance. The ratio of the classification performance (**G-Mean Accuracy**) of the Thr_{smp1} to the optimal adaptive threshold (Thr_{opt}) for the whole of the test data was then computed. This ratio (GAR) was used to measure how close the subset’s cutoff (Thr_{smp1}) was to the optimal one.

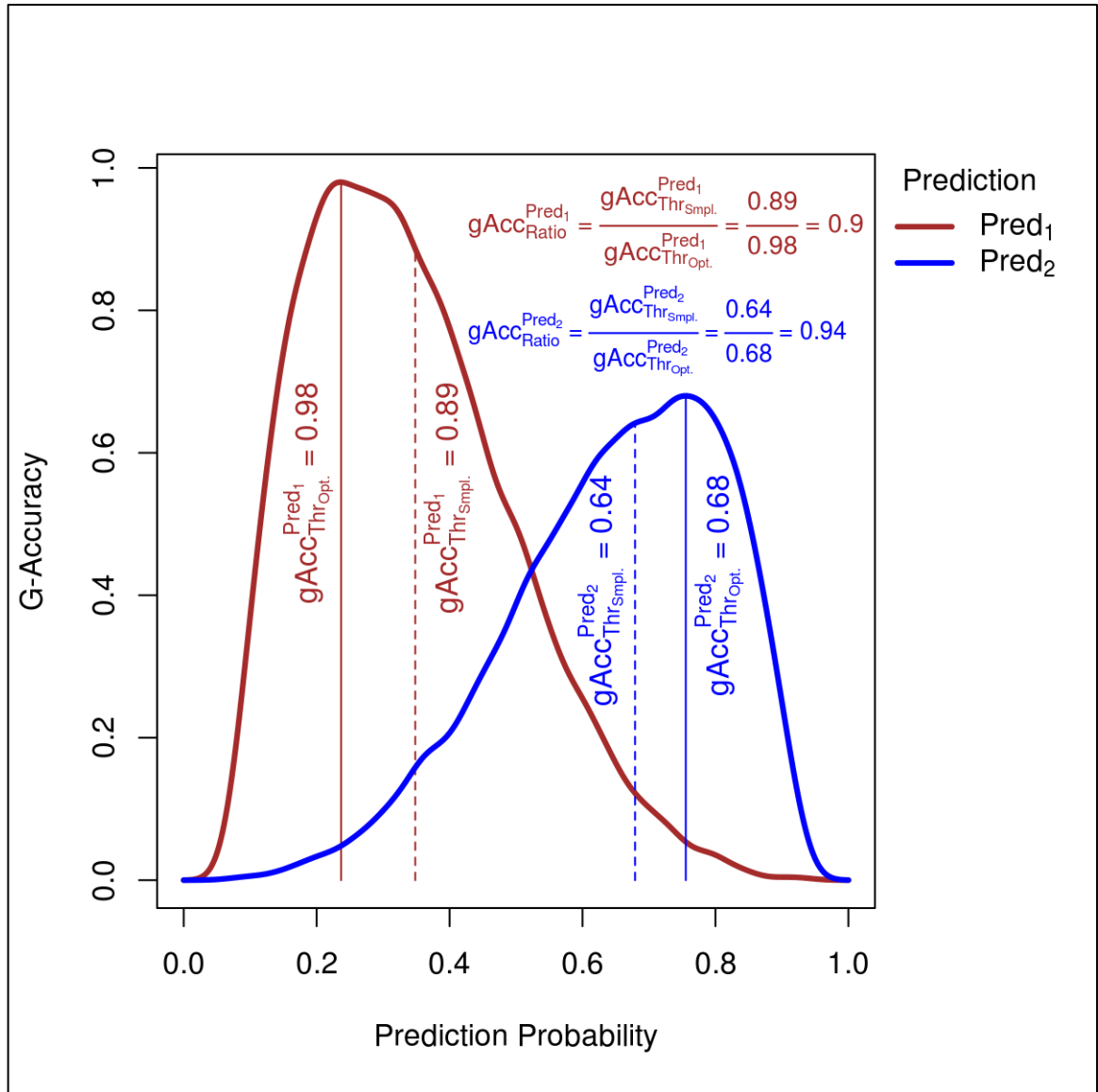


Figure 7.2: Example comparing two predictions (maroon and blue). Vertical solid lines represent the optimal threshold of these predictions and the vertical dotted lines represent the sample cutoffs.

Listing 7.1 presents the pseudo code for the experiments run in these assessments. For every model and evaluation data combination, different samples of various *sizes* were randomly selected. These samples were drawn using *binning* techniques and by randomly introducing different *error* rates to their true labels. Each sample was used to set the prediction threshold (Thr_{smp}), and the quality of this threshold was assessed using the GAR metric. Every sampling process was repeated 100 times and the mean of all of the ratios (GAR) of the 100 repetitions was recorded and used in the later analysis.

Algorithm: Threshold Tuning Experiments**Input:** MDL=Model predictions , testData=The test data the model predicted**Result:** GAR values of sample thresholds (cutoffs)

```

1  Bin.Set          <- {1, 10, 20, 50, 100}
2  Error.Set       <- {0%, 1%, 5%, 10%}
3  SampleSize.Set <- {10%, 5%, 1%, 0.5%, 0.1%, 0.05%, 0.01%, 0.005%, 0.001%, 0.0005%, 0.0001%}
4  Results.Set     <- {}
5
6  For (MDL,testData) results, do           // Process every ML (C5.0, RF, SVM) models'
7                                           // results of the 300 evaluations on
8                                           // the STA2018 dataset
9                                           // (6 Days × 10 Models × 5 test data).
10
11  data.labels <- get.labels(testData)      // Get the true labels of the test data.
12  data.prducts <- get.predictions(MDL,testData) // Get model predictions of the test data.
13  adpt.cutoff <- get.bestCutoff(data.prducts, data.labels) // Get the predictions adapted
14                                           // optimal threshold (Thropt)
15  opt.gAcc <- get.gAcc(data.prducts, data.labels, adpt.cutoff) // Get the G-Mean Accuracy at
16                                           // optimal threshold (Thropt)
17
18  For bin in Bin.Set, do                   // Loop 5 times {1 ... 100}.
19    For err in Errors.Set, do              // Loop 4 times {0% ... 10%}.
20      For smpLSize in SampleSize.Set, do  // Loop 11 times {10% ... 0.0001%}.
21        For repeat in (1..100), do       // repeat every sampling 100 times.
22          // Select random samples and get their labels and predictions
23          i <- select smpLSize instances from every bin of the data.prducts randomly
24          smpL.labels <- data.labels[i]
25          smpL.prducts <- data.prducts[i]
26
27          // Introduce some errors in samples labels randomly
28          if (err > 0)
29            smpL.labels <- swap labels of err instances of smpL.Labels randomly
30
31          // Compute best sample cutoff (ThrsmpL) based on sample labels and predictions
32          if (smpL.labels is pure OR cannot find cutoff)
33            smpL.cutoff <- 0.5
34          else
35            smpL.cutoff <- get.bestCutoff(smpL.prducts, smpL.labels)
36
37          // Compute the G-Mean Accuracy of full data at sample cutoff (ThrsmpL)
38          gAcc <- get.gAcc(data.prducts, data.labels, smpL.cutoff)
39
40          // Compute G-Mean Accuracy Ratio (GAR) of the test data predictions between
41          // sample cutoff and optimal adapted cutoff.
42          gAcc.ratio <- gAcc ÷ opt.gAcc
43
44          // Add results to file for further analysis.
45          Results.Set <- Results.Set U {MDL, testData, bin, err, smpLSize, gAcc.ratio}
46
47          done
48        done
49      done
50    done
51  done
52 done

```

Listing 7.1: Pseudo code of the experiments run for the results of each ML algorithm.

The results from the experiments are illustrated in **Appendix (C)** (**Figure C.19**, **Figure C.20** and **Figure C.21**). Each figure shows the results for the three ML algorithms using different error rates (0%, 1%, 5% and 10%); each subplot shows the median of the GAR values for all of the models with the same feature set and data balance group for that algorithm. The curves

in each subplot for every group illustrate the medians of the different sampling strategies (B_1 , B_{10} , B_{20} , B_{50} and B_{100}) for every sample size (10% to 0.0001%).

Table 7.1 presents the number of sampled instances for every simulation day for every sample size from the transformed STA2018 dataset.

		12-Jun	13-Jun	14-Jun	15-Jun	16-Jun	17-Jun
	Normal	164,545	168,947	213,798	633,388	600,017	409,090
	Attack	2,123	10,037	6,422	35,260	11	4,959
	Total	166,668	178,984	220,220	668,648	600,028	414,049
Sample Size	10%	16,667	17,899	22,022	66,865	60,003	41,405
	5%	8,334	8,950	11,011	33,433	30,002	20,703
	1%	1,667	1,790	2,203	6,687	6,001	4,141
	0.5%	834	895	1,102	3,344	3,001	2,071
	0.1%	167	179	221	669	601	415
	0.05%	84	90	111	335	301	208
	0.01%	17	18	23	67	61	42
	0.005%	9	9	12	34	31	21
	0.001%	2	2	3	7	7	5
	0.0005%	1	1	2	4	4	3
	0.0001%	1	1	1	1	1	1

Table 7.1: Number of sampled instances for each sample size used in these experiments.

7.4 Results and Discussion

These experiments compared the effect of different parameters, i.e. *sample size*, *number of bins* and *error rates*, in determining the optimal threshold (cutoff) for the predictions of the three tested ML algorithms (C5.0, RF and SVM) using the evaluation data. Non-parametric Friedman's tests (with a significance level of $\alpha = 0.05$) were used for these comparisons and the G-Mean Accuracy Ratio (GAR) of the sample cutoffs was used as the evaluation metric. The decision to use non-parametric tests was driven by the non-normal nature of the result GAR values used in these evaluations. The Shapiro-Wilk normality test [332] could not have been used in these experiments due to the size of the experiment results (which exceeded the limit of 5,000 records). Therefore, the Anderson-Darling normality test [333, 334] was used, and confirmed that the results were not normal; $A = 14862.0$, $p = 0.000$.

Due to the number of statistical tests conducted in these analyses, where all of them have shown significant results, only the key hypotheses are explored in the following discussion. The results of their Nemenyi post-hoc tests [373-375] (which were used to perform the pairwise comparisons between different effects to differentiate them) are illustrated with Critical Difference (CD) plots.

Multi-Critical Difference (M-CD) plots are novel illustration that were developed for the comparative analysis in this research and have been used in this chapter. These plots are an aggregation of the multiple CD plots that were used to compare different effects under various conditions.

7.4.1 C5.0 Algorithm

The plots in **Figure 7.3** relate to the C5.0 models. They show the median and interquartile ranges (first and third quartile) of the GAR values for the sample thresholds under different parameters (sample size and number of bins). Every subplot shows the results under different error rates (0%, 1%, 5% and 10%).

As expected, **Figure 7.3** clearly shows that the bigger the sample the better the GAR i.e. that the sample cutoff was the closest to the optimal threshold. However, two unpredicted patterns emerged. Firstly, higher error rates in the sample labels resulted in a better selection of the cutoff for C5.0 predictions. As error rates increased, the GAR values for the larger samples jumped to more than 0.97, while for samples as small as 0.1%, values crossed the 0.9 GAR point at the 10% error rate. Secondly, the B_1 sampling strategy was superior to other sampling approaches, especially for sample sizes of less than 5%. This advantage was maintained across increased error rates with a widening gap between B_1 and the multi-bin sampling strategies at lower sampling rates.

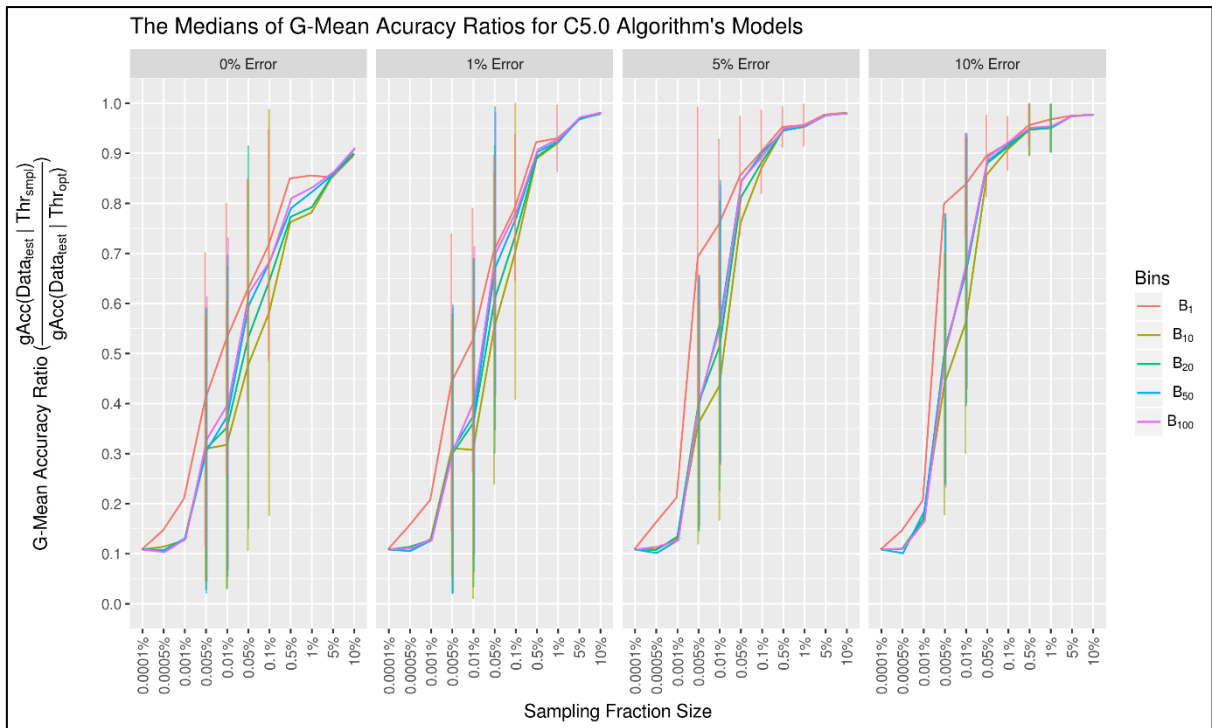


Figure 7.3: Median of G-Mean Accuracy Ratios (GAR) of the C5.0 models predictions under different sampling strategies (number of bins, sample sizes and error rates).

The cause of such unexpected behaviours is linked to the nature of C5.0 models, which produced low numbers of unique probabilities (thresholds) for their predictions. **Figure 7.4** shows the averages of the number of unique probability values (thresholds) returned by every C5.0 models (generated in **Chapter 6**) as predictions of the tested data. The blue and red lines respectively represent the minimum and maximum probabilities (thresholds) within the range of these predictions. This figure shows that the predictions of the C5.0 models suffered from having low numbers of unique cutoffs and short ranges. In general, C5.0 models produced far fewer thresholds than RF and SVM. This caused a shift from one cutoff value to another -in C5.0 predictions- leading to a step jump which caused many samples to change state in a single step.

In the RF and SVM predictions these transitions were much smoother due to the high number of prediction probabilities (thresholds).

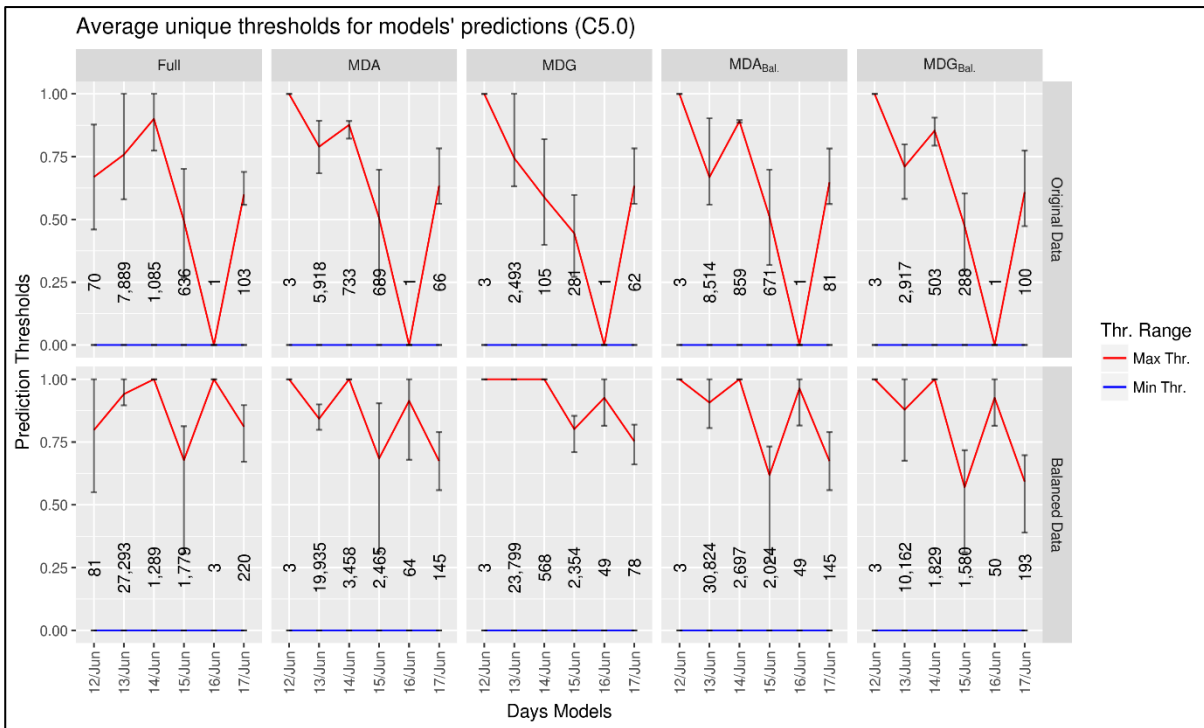


Figure 7.4: Average number of unique thresholds for the predictions of C5.0 models, and their ranges.

As a result of the low number of unique thresholds produced by the C5.0 models, the random samples tended to include all of the cutoff values returned by these models. Therefore, the mixed cases (of normal and attack instances) tended to skew the selected cutoff (Thr_{Smp}) away from the optimal threshold (Thr_{Opt}) and towards those rarer minority classes (attack) to maximise the performance measure (G-Mean Accuracy). However, as the error rates increased, the likelihood of changing the labels of majority instances (normal) will be higher, causing the sample cutoff (Thr_{Smp}) to shift towards the optimal threshold (Thr_{Opt}). This resulted in a big improvement in their GAR values due to the effect of those step jumps. This effect is discussed in more detail in **Section 7.4.4** and illustrated in **Figure 7.15-(a)** (see **Page 205**).

Although the GAR curves in **Figure 7.3** suggest the B₁ sampling strategy was superior to the multi-bin strategies, statistical analysis was conducted to test the significance of this observation.

To statistically analyse the differences between various sampling strategies (number of bins) for the identification of close to optimal threshold, four Friedman's tests were performed with

a significance level of $\alpha=0.05$. Each test was performed under a different error rate to test the hypothesis, “*there are no statistically significant differences in the G-Mean Accuracy Ratios (GAR) of the sample cutoffs of the C5.0 model predictions between the different sampling strategies –number of bins- (B_1 , B_{10} , B_{20} , B_{50} and B_{100}) when the sample labels have an X% error rate*”. These are the results for each test:

- 0% Error : $\chi^2(4) = 411.3$, $p = 0.000 < 0.05$ (*differences are statistically significant*)
- 1% Error : $\chi^2(4) = 304.5$, $p = 0.000 < 0.05$ (*differences are statistically significant*)
- 5% Error : $\chi^2(4) = 295.0$, $p = 0.000 < 0.05$ (*differences are statistically significant*)
- 10% Error : $\chi^2(4) = 247.0$, $p = 0.000 < 0.05$ (*differences are statistically significant*)

Every one of these tests showed significant differences between the sampling strategies where a Nemenyi post-hoc test had been conducted to perform a pairwise comparison as illustrated in the M-CD plots in **Figure 7.5**.

This plot (**Figure 7.5**) shows that B_1 has no significant differences to both B_{50} and B_{100} when the error rate was between 0% and 5%. However, once the error rate reached 10%, the differences between most sampling strategies became statistically significant, except for B_{50} and B_{100} , where B_1 remained the best.

Further investigation into the possible cause of this unpredictable result i.e. the superiority of the B_1 sampling strategy, revealed that the models’ predictions for Day 2 (12/Jun) and Day 6 (16/Jun) were the most problematic as a result of them having a very small number of unique thresholds (see **Figure 7.4**). This led to a large skewness of Thr_{Smpl} when the multi-bins strategies are applied due to the presence of many extreme thresholds that were missed by the B_1 approach. The predictions for these two days skewed the overall results. However, the Friedman test was able to detect the insignificance of this effect, to some extent. The results for these two days are set out in **Figure C.1** and **Figure C.5** in **Appendix (C)** [the reader is referred to **Figure B.1** for further details about the performance plots for C5.0 models for Day 2 and Day 6 in **Appendix (B)**].

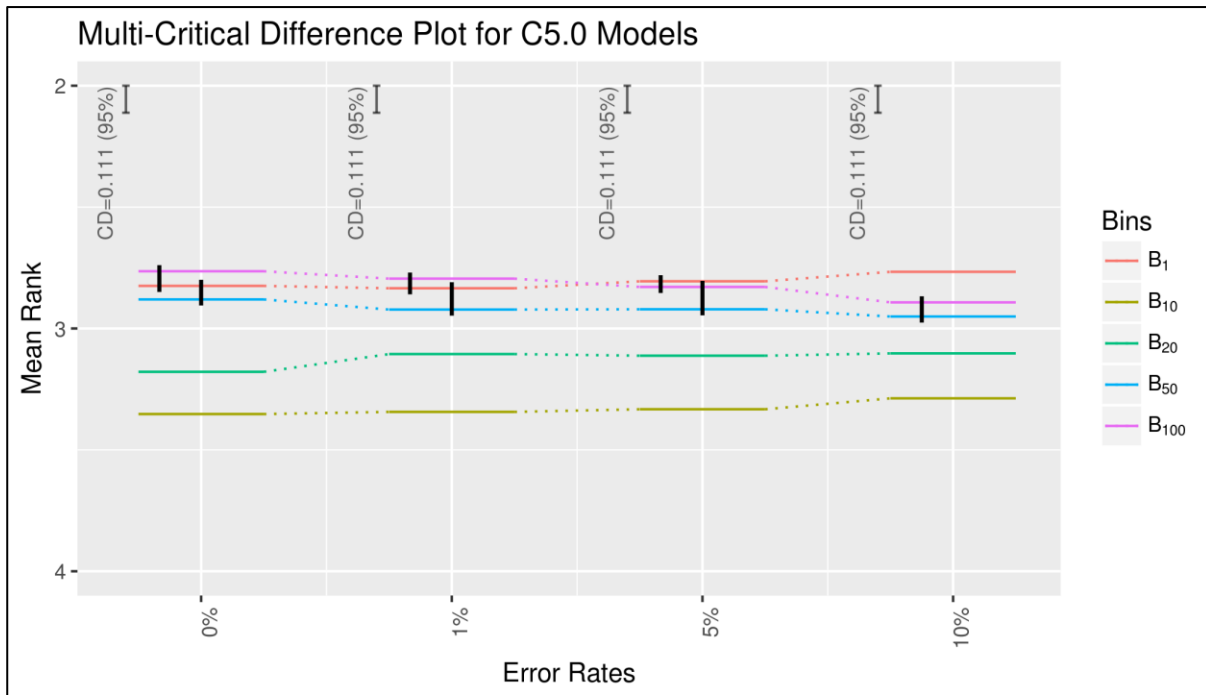


Figure 7.5: Results of multiple Nemenyi tests (95% confidence level) on different sampling strategies (B_1 , B_{10} , B_{20} , B_{50} and B_{100}) using the C5.0 predictions under different error rates (0%, 1%, 5% and 10%).

To analyse the effect of different sample sizes on the accuracy of sample cutoffs under various error rates, four Friedman tests were performed with a significance level of $\alpha=0.05$. Each tested the following hypothesis, “*there are no statistically significant differences in the G-Mean Accuracy Ratios (GAR) of the sample cutoffs of C5.0 model predictions between the different sample sizes (10% to 0.0001%) when the sample labels have an X% error rate*”. The results from these tests are as follow:

- 0% Error : $\chi^2(10) = 4983.6$, $p = 0.000 < 0.05$ (*differences are statistically significant*)
- 1% Error : $\chi^2(10) = 7470.6$, $p = 0.000 < 0.05$ (*differences are statistically significant*)
- 5% Error : $\chi^2(10) = 7162.4$, $p = 0.000 < 0.05$ (*differences are statistically significant*)
- 10% Error : $\chi^2(10) = 6729.7$, $p = 0.000 < 0.05$ (*differences are statistically significant*)

All of these tests showed significant differences. **Figure 7.6** illustrates the M-CD plots of the pairwise comparisons subsequently conducted using the Nemenyi post-hoc test for every error rate.

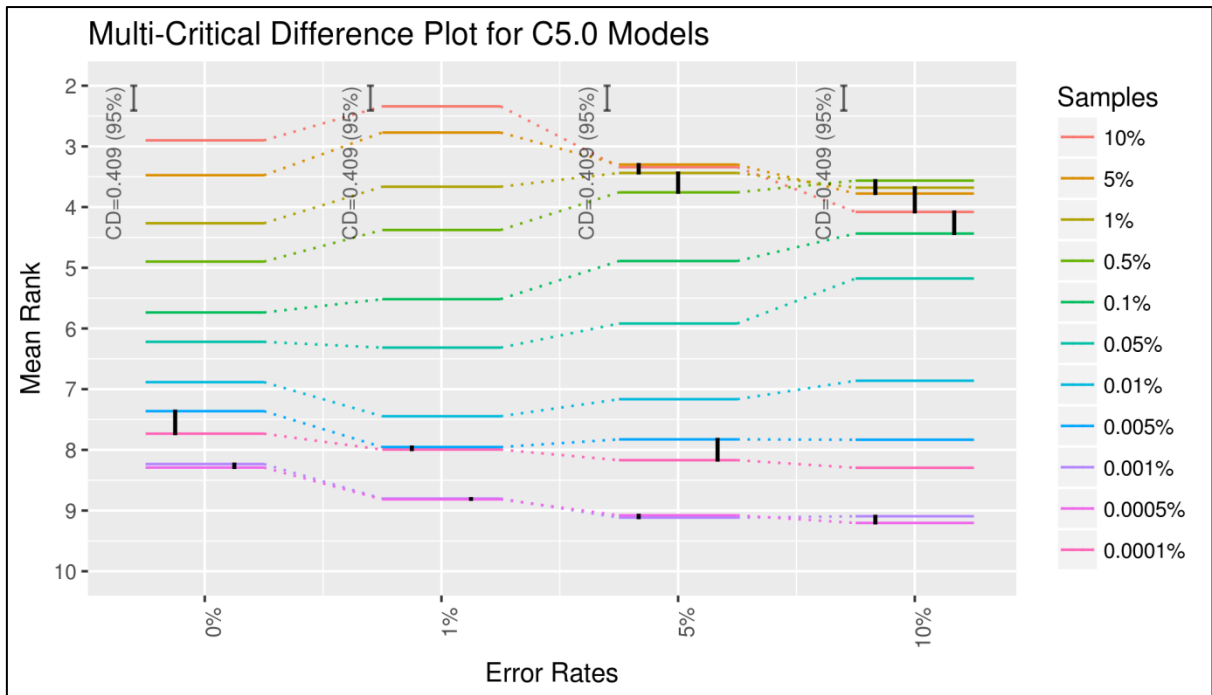


Figure 7.6: Results of multiple Nemenyi tests (95% confidence level) of different sampling size (10% to 0.0001%) using the C5.0 predictions under different error rates (0%, 1%, 5% and 10%).

These tests showed significant differences between the sample sizes; the bigger the sample, the higher the ranking it achieved, confirming the observations set out in **Figure 7.3**. For the smaller samples (less than 0.05%), the differences were insignificant. Moreover, as the error rate in sample labels reached 5%, the difference between the sample sizes of 10%, 5%, 1% and 0.5% became insignificant, including the sample size of 0.1% at an error rate of 10%. The smaller the sample size, the less representative they were of the original population (test data); this caused the cutoff Thr_{Smp1} to be selected, which is far from the optimal Thr_{Opt} . The effect of increasing the error rates for the larger samples is discussed in more detail in **Section 7.4.4**.

7.4.2 RF Algorithm

Figure 7.7 presents the medians and interquartile ranges for the GAR values of the samples' cutoffs for the RF predictions. The subplots show the results of different sample sizes and the number of bins at every error rate (0%, 1%, 5% and 10%).

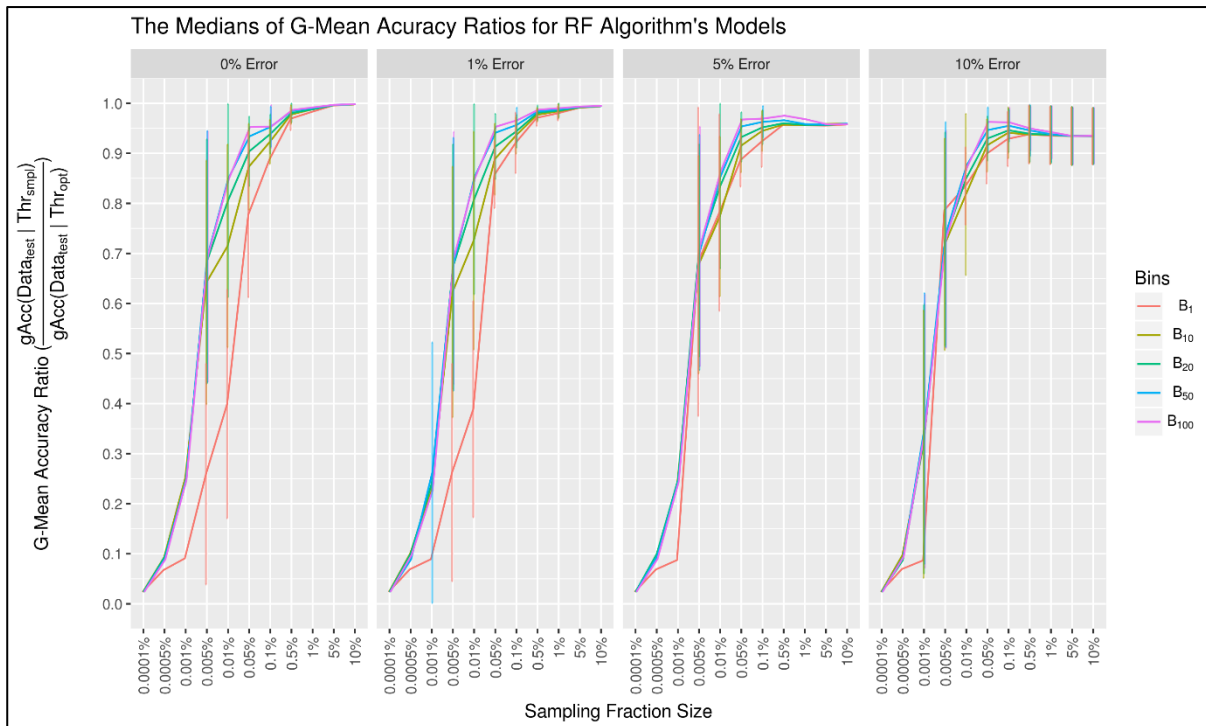


Figure 7.7: Median of G-Mean Accuracy Ratios (GAR) of the RF models predictions under different sampling strategies (number of bins, sample sizes and error rates).

As with the C5.0 predictions, the cutoffs for the larger samples were much better, as they had higher GAR values - as can be seen in **Figure 7.7**. However, unlike C5.0, the GAR of the RF samples dropped as the error rate increased, especially amongst the larger samples (10% to 0.5%). In general, sample sizes below 0.05%, lost their accuracy in selecting close to optimal cutoffs with their GAR starting to rapidly fall below 0.9. The key difference between the behaviour of C5.0 and RF is linked to the large number of unique thresholds produced by the RF models, which are illustrated in **Figure 7.8**. For this reason, a larger sample size will have better representation of the original data and will manifest a wider and finer prediction range with a proportionate number of class labels close to the real data. This led to the selection of better cutoffs that were close to the optimal threshold. (Further explanations to explore the effect of the errors are discussed in **Section 7.4.4**.)

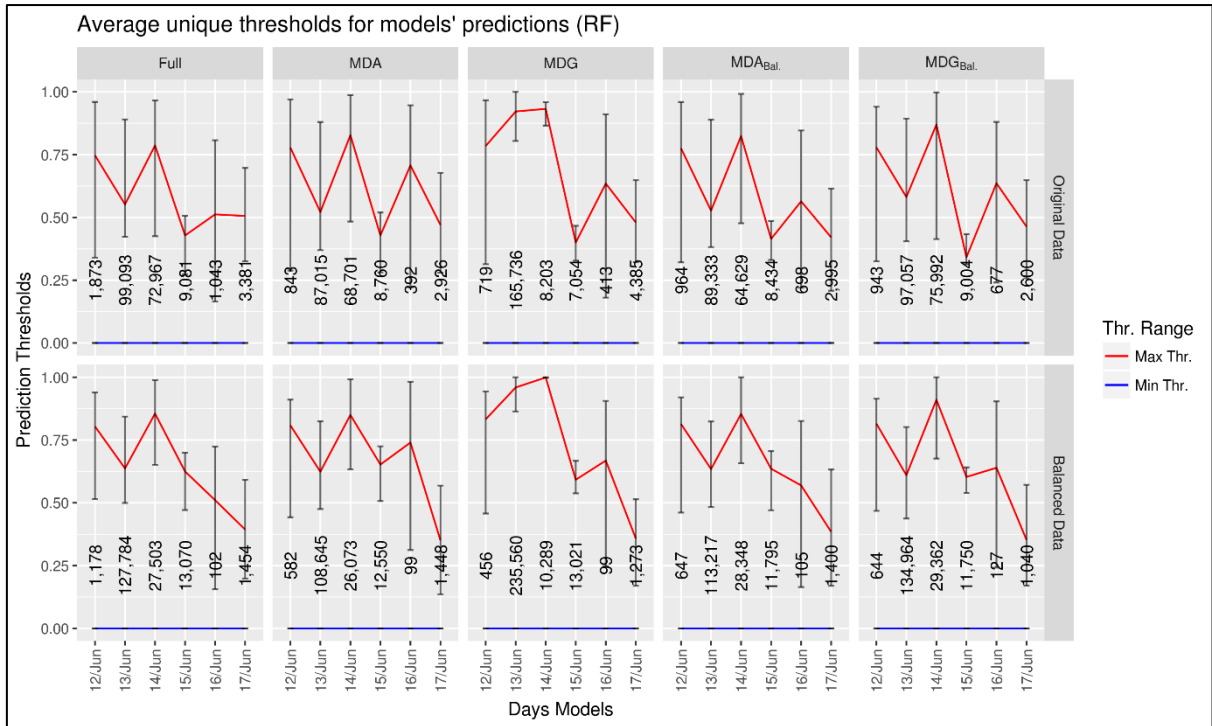


Figure 7.8: Average number of unique thresholds for the predictions of the RF models, and their ranges.

Moreover, as initially anticipated, the B_1 sampling strategy exhibited a lower GAR than the multi-bin sampling approach and these differences were much more obvious amongst the smaller samples. It was also noticeable that the gap between the B_1 and the multi-bin sampling strategies narrowed as the error rate increased. These findings were further analysed to determine their significance.

Different sampling strategies (number of bins) were analysed to determine if they have any differences in finding the best threshold based on their sample cutoffs. Four Friedman tests were executed with a significance level of $\alpha=0.05$. Tests were performed for each error rate to test the following hypothesis, “*there are no statistically significant differences in the G-Mean Accuracy Ratios (GAR) of the sample cutoffs for the RF model predictions between the different sampling strategies –number of bins- (B_1 , B_{10} , B_{20} , B_{50} and B_{100}) when the sample labels have an X% error rate*”. The results from the tests are as follows:

- 0% Error : $\chi^2(4) = 1499.0$, $p = 0.000 < 0.05$ (*differences are statistically significant*)
- 1% Error : $\chi^2(4) = 1181.9$, $p = 0.000 < 0.05$ (*differences are statistically significant*)
- 5% Error : $\chi^2(4) = 900.4$, $p = 0.000 < 0.05$ (*differences are statistically significant*)
- 10% Error : $\chi^2(4) = 703.9$, $p = 0.000 < 0.05$ (*differences are statistically significant*)

Figure 7.9 shows the M-CD plots of the Nemenyi post-hoc test, which was subsequently conducted to identify any differences between the sampling strategies by performing pairwise comparisons. These plots show that the larger the number of bins, the better the sample cutoff. The differences between all of the sampling strategies were significant with the larger bins ranked higher than the smaller ones.

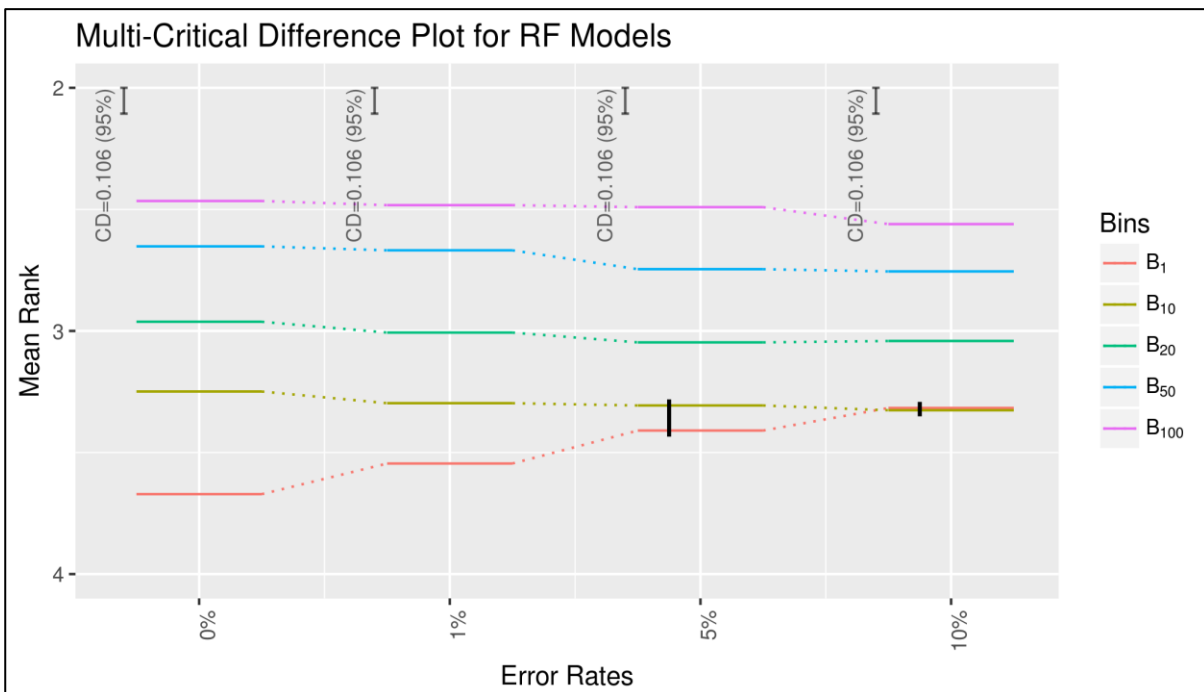


Figure 7.9: Results of multiple Nemenyi tests (95% confidence level) on different sampling strategies (B_1 , B_{10} , B_{20} , B_{50} and B_{100}) using the RF predictions under different error rates (0%, 1%, 5% and 10%).

This confirms the observations set out in **Figure 7.7** which showed that the curve of the B_1 strategy was lower than that of the other sampling strategies. As error rates reached 5% and above, the difference between B_1 and B_{10} became insignificant. This was due to the increase in the GAR values of the B_1 samples, which reduced the gap (between the B_1 and B_{10} lines), as illustrated in **Figure 7.7**.

This result is in line with what was expected as a wider prediction range should increase the level of accuracy of identifying the right discriminating threshold. The key difference between the C5.0 predictions and those of RF and SVM was the number of unique thresholds produced, as discussed earlier. As the number of unique cutoffs decreased in the C5.0 predictions, finding the optimal threshold using a larger number of bins, became less efficient.

Four Friedman tests were performed (with a significance level of $\alpha=0.05$) to analyse the effect of different sample sizes on the selected sample cutoffs under various error levels. Each test assessed the following hypothesis, “*there are no statistically significant differences in the G-Mean Accuracy Ratios (GAR) of the sample cutoffs of the RF model predictions between the different sample sizes (10% to 0.0001%) when the sample labels have an X% error rate*”.

The results from these tests are as follow:

- 0% Error : $\chi^2(10) = 12052.0$, $p = 0.000 < 0.05$ (*differences are statistically significant*)
- 1% Error : $\chi^2(10) = 11024.0$, $p = 0.000 < 0.05$ (*differences are statistically significant*)
- 5% Error : $\chi^2(10) = 8799.4$, $p = 0.000 < 0.05$ (*differences are statistically significant*)
- 10% Error : $\chi^2(10) = 8472.3$, $p = 0.000 < 0.05$ (*differences are statistically significant*)

All of these tests showed significant statistical differences. **Figure 7.10** illustrates the M-CD plots of the pairwise comparisons that were subsequently conducted using the Nemenyi post-hoc test.

Like the C5.0 algorithm plots, these plots (**Figure 7.10**) show significant differences between different sample sizes. At error rates of 0% and 1%, the larger the sample size, the higher the ranking. As the error rate reached 5%, some of the small samples (0.5%, 0.1% and 0.05%) started to become indistinguishable from the larger samples, but once the error rate reached 10%, these small samples became highly ranked and their difference to the larger sample sizes become significant. However, the smaller samples (less than 0.05%) were always ranked low.

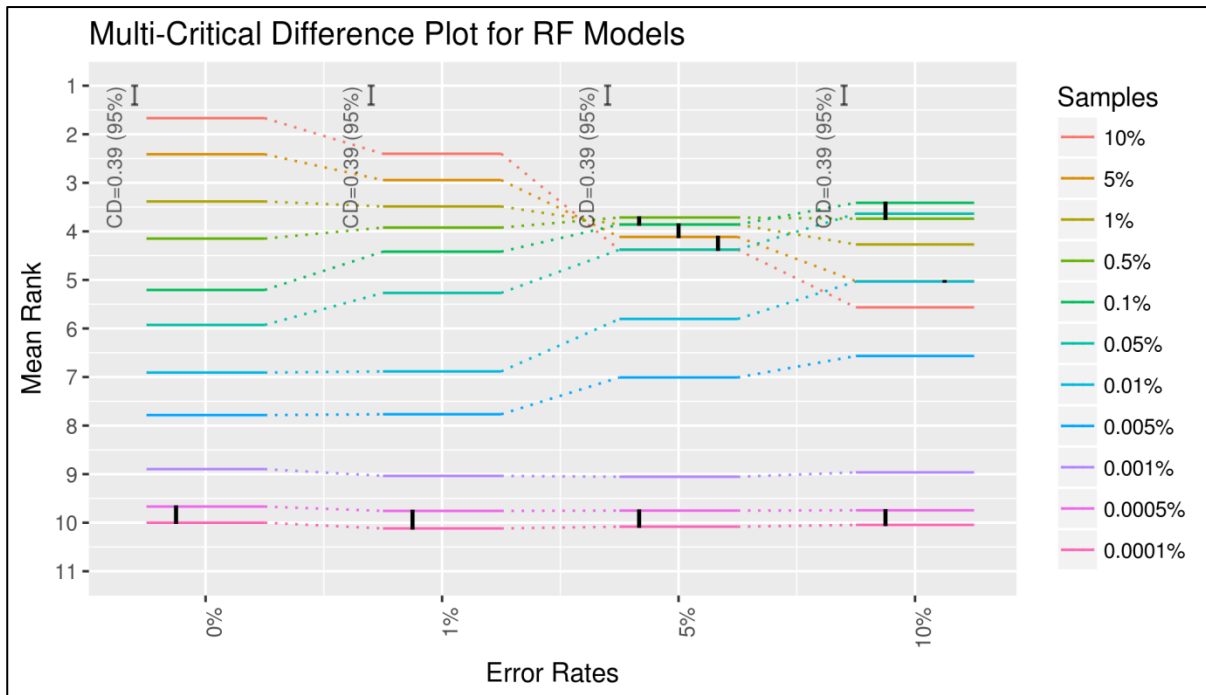


Figure 7.10: Results of multiple Nemenyi tests (95% confidence level) on different sampling sizes (10% to 0.0001%) using **RF** predictions under different error rates (0%, 1%, 5% and 10%).

For these small samples (0.5%, 0.1% and 0.05%) the number of unique thresholds were small, resulting in step jump effects similar to the C5.0 predictions discussed in **Section 7.4.4**. This led to the selection of Thr_{Smpl} that is furthest from Thr_{Opt} . As a result, high error rates caused many of the majority (normal) instances to change their state, which in turn caused the sample cutoffs (Thr_{Smpl}) to shift towards the optimal threshold. This can be seen from the increase in their GAR and their statistical rankings as illustrated in **Figure 7.15-(a)**. However, as the larger samples had already selected an accurate Thr_{Smpl} (which is the closest to the Thr_{Opt}), higher error rates resulted in a skewness of this Thr_{Smpl} towards these erroneous cases away from the optimal threshold, which in turn resulted in a fall in their GAR values. This made any differences in these samples indistinguishable from those of the small samples [see **Figure 7.15-(a)**].

7.4.3 SVM Algorithm

As for the SVM predictions, **Figure 7.11** illustrates the medians for the sample GAR results and their interquartile ranges (first and third quartile). These subplots show the result of

different parameters (sample size and the number of bins) for every error rate (0%, 1%, 5% and 10%).

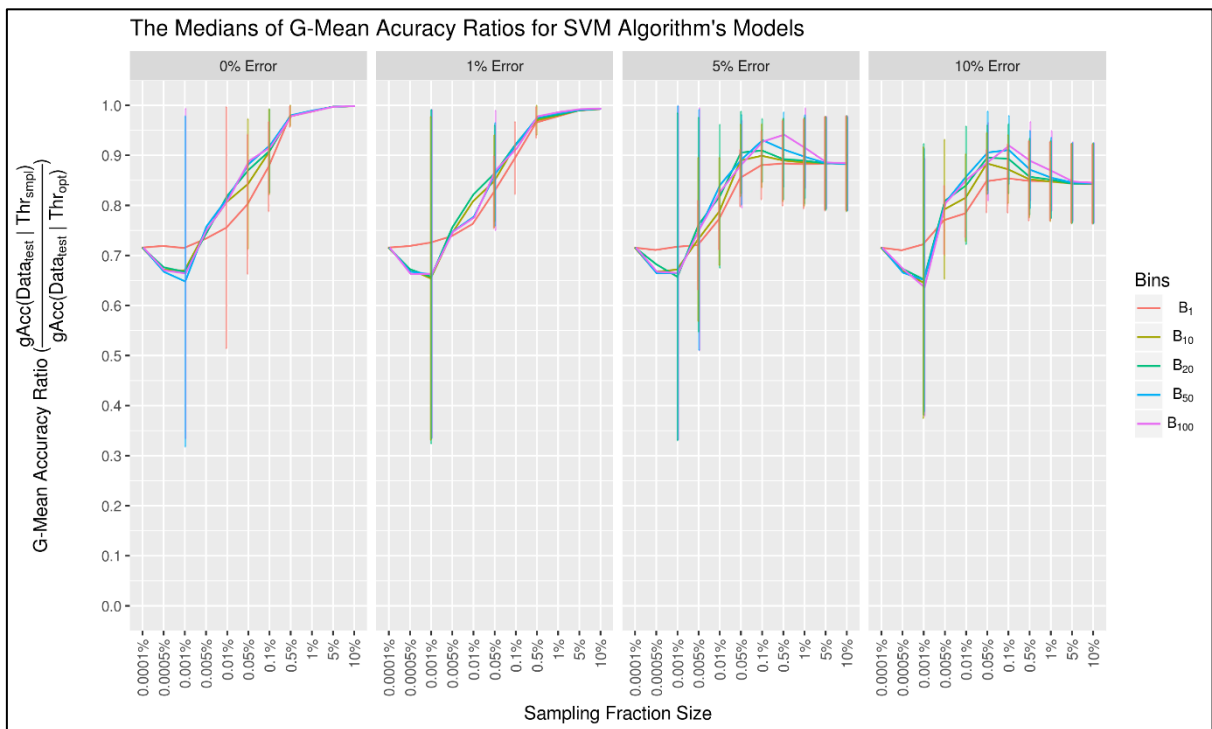


Figure 7.11: Median of G-Mean Accuracy Ratios (GAR) of the predictions of the SVM models under different sampling strategies (number of bins), sample sizes and error rates.

Like the C5.0 and RF predictions, the SVM predictions were better at estimating the optimal threshold when larger sample sizes were used, as illustrated in **Figure 7.11**. The GAR of the SVM samples showed a similar trend to the RF samples in terms of error rates. However, the SVM samples showed a higher sensitivity to errors as their GAR values declined more sharply when error rates increased. By the time the error rates reached 5% and 10% the GAR fell below 0.9 for the larger sample sizes (10% to 0.5%). This was caused by the large number of unique probabilities (thresholds) produced by the SVM models, which were much larger than those produced by the C5.0 and RF models, as can be seen in **Figure 7.12**. There were nearly as many of these thresholds as there were instances (connections) in the evaluation data. In other words, nearly every predicted instance had its own unique probability. Furthermore, these predictions covered the full range between zero and one with very few fluctuations, unlike the C5.0 and RF

predictions. The effect of errors on the samples' cutoff selection is discussed in more detail in

Section 7.4.4.

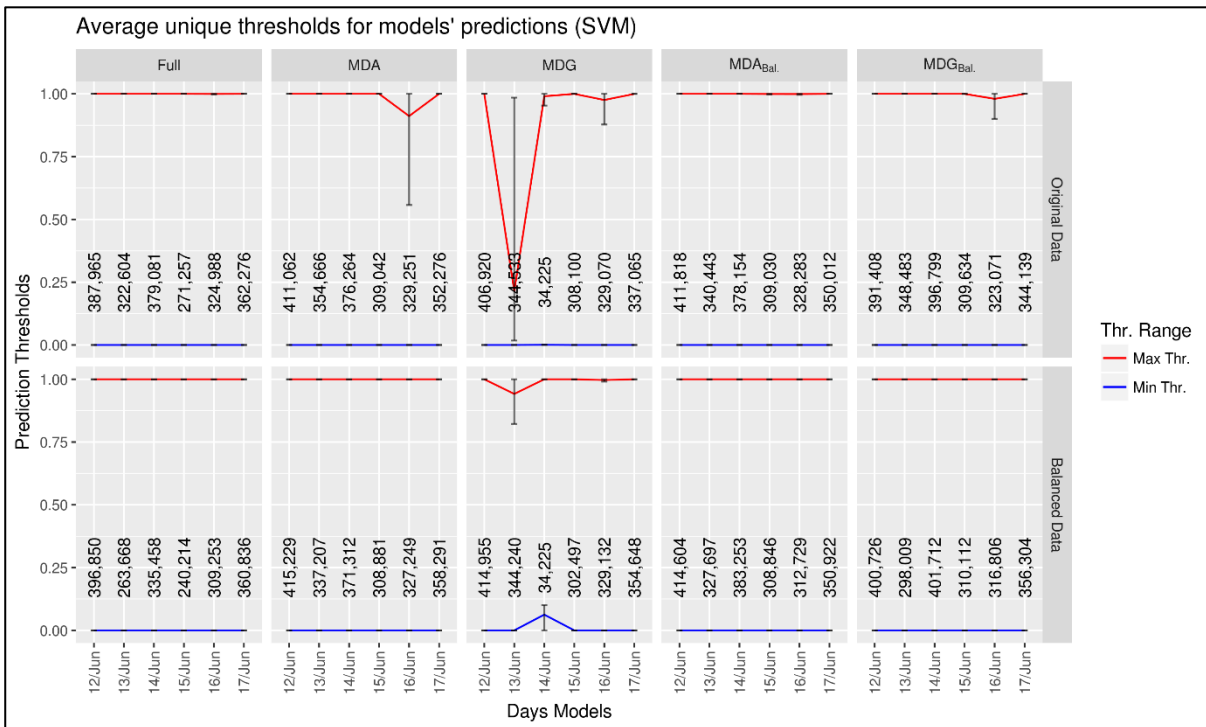


Figure 7.12: Average number of unique thresholds for the predictions of the SVM models and their ranges.

In a pattern similar to the RF predictions, the GAR values for the SVM samples using a multi-bin sampling strategy, tended to be higher than the B₁ sampling as can be seen in **Figure 7.11**. However, initially an unexpected pattern emerged - the SVM samples showed a higher GAR value for small sample sizes (0.05% to 0.0001%) compared to those for C5.0 and RF. Also, for the smaller samples ($\leq 0.005\%$), the B₁ sampling strategy had higher and more stable GAR values. This was down to the fact that for these smaller samples, the chances of a pure sample (single label) being selected was very high. As a result, Thr_{Smp1} could not be identified which led to the sample being assigned the default cutoff value of 0.5 (Thr_{0.5}) as shown in **Listing 7.1 (lines 32-35)**.

In the multi-bin sampling strategies, the width of the prediction range was taken into consideration, which resulted in cases being sampled from every interval, which in turn led to more mixed samples. This resulted in the selection of a cutoff that was far from the optimal

one. To determine the significance of these effects and to explore their contributing factors, further statistical analysis was undertaken.

The reason for the high GAR values for the smaller SVM samples (0.05% to 0.0001%) compared to those of the C5.0 and RF samples is related to the nature of the G-Accuracy curve for the SVM predictions. Most of the SVM prediction ranges were wide (0-1) and the curvature of their G-Accuracy curves is much flatter in the middle. Setting the sample cutoff (threshold) for a small sample size to a default of 0.5 ($\text{Thr}_{0.5}$) did not result in a greater decrease in the GAR rate. This is because a threshold of 0.5 is within the range of most SVM predictions. This is not the case for most of the C5.0 and RF predictions, where a threshold of 0.5 ($\text{Thr}_{0.5}$) lies at the edge or outside the range of their predictions as they form shorter intervals. These issues resulted in very low GAR values that can reach zero in some cases at the default threshold of 0.5 ($\text{Thr}_{0.5}$).

These cases are illustrated in **Figure 7.1**, where the G-Accuracy curve of the SVM predictions are similar to the Pred_3 curve while the G-Accuracy curves of the C5.0 and RF predictions would take the shape of Pred_1 or Pred_2 .

An analysis of the different sampling strategies (the number of bins) was conducted to establish whether there was any difference between the strategies in estimating the best threshold. A Friedman test was performed (with a significance level of $\alpha=0.05$) for each error rate, to test the hypothesis, “*there are no statistically significant differences in the G-Mean Accuracy Ratios (GAR) of the sample cutoffs of the SVM model predictions between the different sampling strategies –number of bins- ($B_1, B_{10}, B_{20}, B_{50}$ and B_{100}) when the sample labels have an $X\%$ error rate*”. The results of the tests are as follows:

- 0% Error : $\chi^2(4) = 29.6$, $p = 0.000 < 0.05$ (*differences are statistically significant*)
- 1% Error : $\chi^2(4) = 308.3$, $p = 0.000 < 0.05$ (*differences are statistically significant*)
- 5% Error : $\chi^2(4) = 666.6$, $p = 0.000 < 0.05$ (*differences are statistically significant*)
- 10% Error : $\chi^2(4) = 764.9$, $p = 0.000 < 0.05$ (*differences are statistically significant*)

The M-CD plots in **Figure 7.13** illustrate the results from the pairwise comparisons subsequently generated by a Nemenyi post-hoc test, which was used to identify the differences between the various sampling strategies.

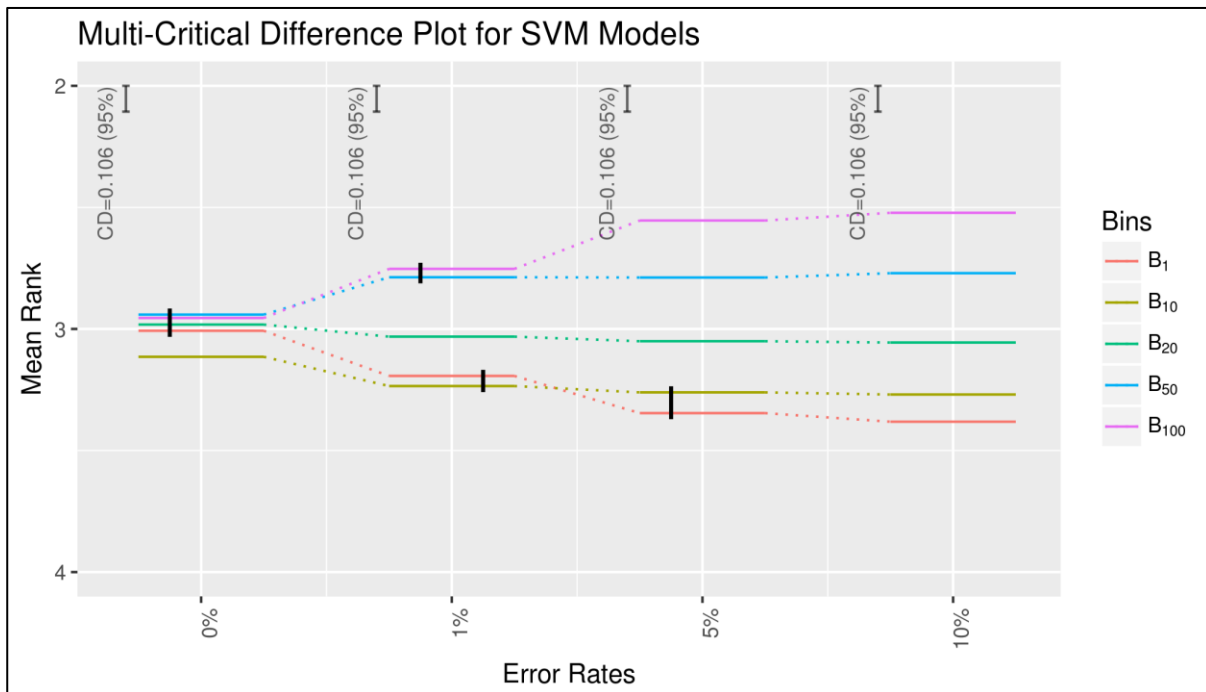


Figure 7.13: Results of multiple Nemenyi tests (95% confidence level) on different sampling strategies (B_1 , B_{10} , B_{20} , B_{50} and B_{100}) using the SVM predictions under different error rates (0%, 1%, 5% and 10%).

Figure 7.13 shows that for the SVM models (unlike the C5.0 and RF models) there were insignificant differences between the different sampling strategies when the sample labels were accurate (0% error), except in the case of the B_{10} sampling approach which was ranked the lowest. As the error rate increased, the differences between the sampling strategies became more significant with the multi-bin sampling methods starting to become highly ranked. When the error rate reached 10%, B_1 was ranked the lowest, while the higher the number of bins, the better the ranking.

There were two main reasons for this. The first was an increase in the GAR values for B_1 . Very small sample sizes caused an increase in the ranking by the statistical test. The second reason was the large number of unique thresholds in the SVM predictions. Unlike C5.0 and RF, when the B_1 strategy was applied to the SVM predictions, sample instances did not concentrate at

particular thresholds. As a result, the samples produced had a wider coverage of the prediction range, which in turn led to a better selection of the sample threshold Thr_{Smp1} (which is very close to Thr_{Opt}). However, the middle sized samples (1%-0.005%) produced a similar effect to the C5.0 and RF samples in that the selected samples would have a small number of thresholds. This was due to the low number of instances sampled, resulting in a step (jump) transition effect between the thresholds. As the error rate for these samples increased, the GAR for the selected sample threshold (Thr_{Smp1}) increased and different sampling strategies started to differentiate from each other. The statistical test took these effects into account and gave a higher ranking to the multi-bin strategies. In these cases, the larger multi-bins had better GAR values as they offered better coverage of the prediction range.

The analysis was extended to examine the impact of sample size on the selected sample cutoffs under various error levels, using the Friedman test (with a significance level $\alpha=0.05$). Four tests were undertaken to test the following hypothesis, *“there are no statistically significant differences in the G-Mean Accuracy Ratios (GAR) of the samples’ cutoffs of the SVM model predictions between the different sample sizes (10% to 0.0001%) when the sample labels have an X% error rate”*. The results of these tests were as follows:

- 0% Error : $\chi^2(10) = 7221.3$, $p = 0.000 < 0.05$ (*differences are statistically significant*)
- 1% Error : $\chi^2(10) = 6226.3$, $p = 0.000 < 0.05$ (*differences are statistically significant*)
- 5% Error : $\chi^2(10) = 3515.0$, $p = 0.000 < 0.05$ (*differences are statistically significant*)
- 10% Error : $\chi^2(10) = 2800.3$, $p = 0.000 < 0.05$ (*differences are statistically significant*)

All of these tests showed significant differences. **Figure 7.14** illustrates the M-CD plots of the pairwise comparisons subsequently generated using the Nemenyi post-hoc test.

For SVM, **Figure 7.14** shows that the differences between the different sample sizes were statistically significant. As with RF, at low error rates (0% and 1%), the larger the sample size, the higher the ranking. However, the small sample sizes (0.5%, 0.1% and 0.05%) started to

show superiority over larger samples (10%, 5% and 1%), with significant differences as the error rates increased to 5% and beyond.

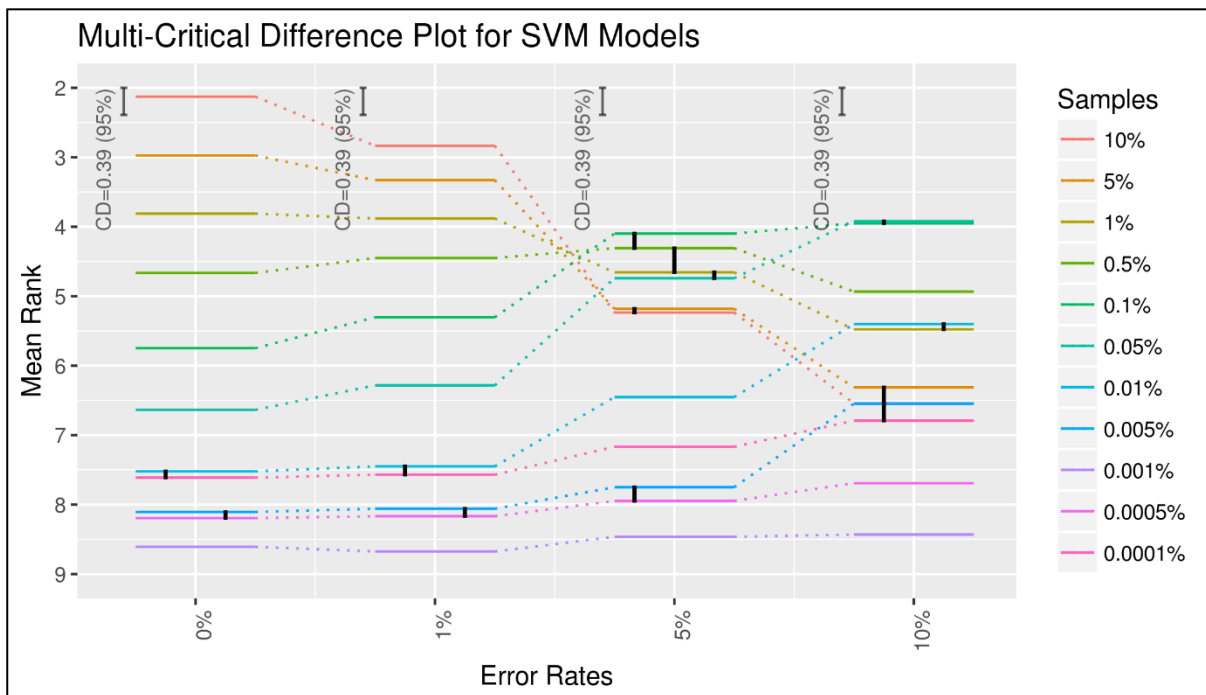


Figure 7.14: Results of multiple Nemenyi tests (95% confidence level) on different sampling sizes (10% to 0.0001%) using the SVM predictions under different error rates (0%, 1%, 5% and 10%).

As with the RF predictions, two significant but opposite effects began to happen at the same time: while there was a reduction in the GAR values of the larger samples, the GAR values of the small samples (0.5%, 0.1% and 0.05%) increased. These small samples had a few number of unique thresholds, resulting in the step jump effect discussed earlier.

7.4.4 Closing remarks

Figure 7.15 illustrates the effect of error rates (in the sample labels) on the selection of sample thresholds for the different ML models.

The experiments outlined in Chapter 6 were configured to return the probability of an instance (connection) being an attack. Therefore, the lower the probability (of an attack) the higher the chance that the evaluated connection was normal. As a result, a higher density of normal class connections were present at the lower end of the prediction scale. Moreover, introducing errors

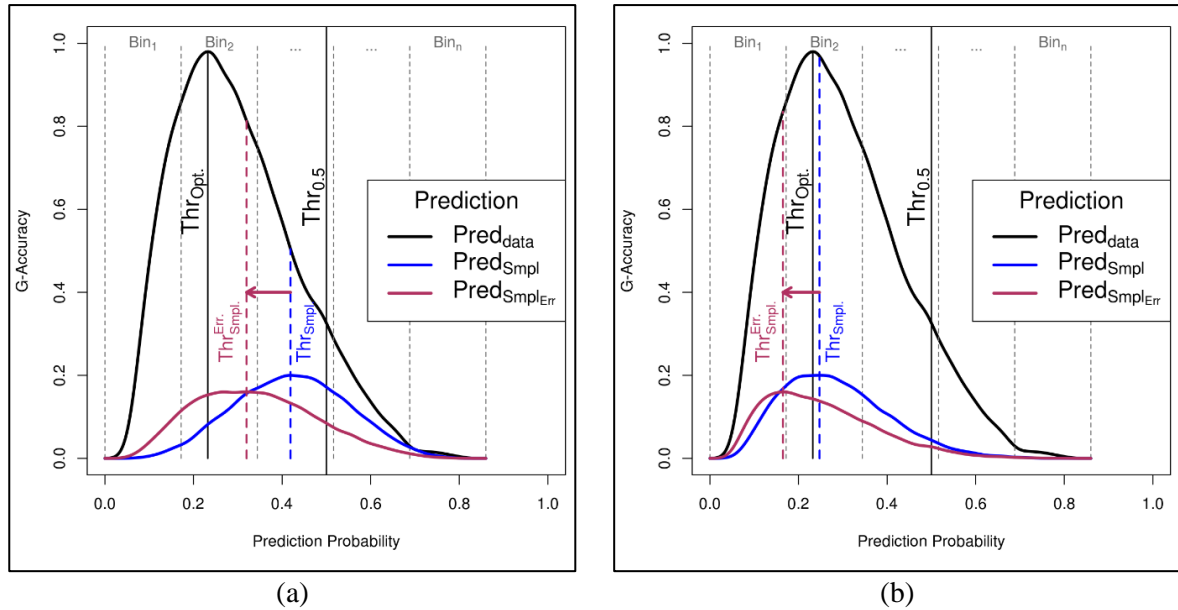


Figure 7.15: Illustrative plots of the effect of erroneous sample labels on threshold shift (a) C5.0 predictions (b) RF and SVM predictions.

randomly into these sample labels changed the labels of many of these normal instances into an attack, resulting in a shift of the threshold towards erroneous cases to maximize the performance measure (G-Mean Accuracy) as explained earlier. **Figure 7.15** illustrates the changes in the threshold under different scenarios. **Figure 7.15-(a)** shows the cases behaved in a similar way to the C5.0 predictions, where the sample cutoff (Thr_{Smpl}) would usually be far from the optimal threshold (Thr_{Opt}), and errors in the sample labels resulted in a shift of the cutoff towards Thr_{Opt} . In the RF and SVM predictions, the opposite effect happened [see **Figure 7.15-(b)**] as the sample cutoff (Thr_{Smpl}) would originally be the closest to the optimal threshold (Thr_{Opt}) due to the sample's good representation of the original data. In such cases, increasing the error rate introduced a shift that is similar to the previous case, though this shift pushed the sample cutoff (Thr_{Smpl}) away from the optimal threshold (Thr_{Opt}) which resulted in a fall in the GAR.

There are two key factors that affected the main behaviour of these algorithms' predictions in identifying the right threshold, based on the labels of small samples: *the number of unique predictions* (probabilities) generated by the models; and *the predictions range*.

Overall, in relation to sample size all of the algorithms behaved similarly in determining the ‘close to optimal’ threshold i.e. the larger the sample, the better the estimation. Almost all of these ML algorithms also displayed the same behaviour in relation to those smaller samples (i.e. smaller than 0.01%) that were ranked the lowest (as illustrated in **Figure 7.6**, **Figure 7.10** and **Figure 7.14**). Therefore, the following discussion and analyses will focus on sample sizes greater than 0.005% (i.e. 10%, 5%, 1%, 0.5%, 0.1% 0.05% and 0.01%).

A final comparison looked at the algorithms’ predictions under different error levels to determine the best cutoff based on a small subset. For this, a Friedman test was performed (with a significance level of $\alpha=0.05$) to test the following hypothesis, “*there are no statistically significant differences in the G-Mean Accuracy Ratios (GAR) of the sample cutoffs of the different ML (C5.0, RF and SVM) model predictions at different error rates (0%, 1%, 5% and 10%) in the sample labels*”.

As the results showed significant differences - $\chi^2(11) = 18417.0$, $p = 0.000 < 0.05$ - pairwise comparisons were performed using the Nemenyi post-hoc test. The results of the test are illustrated in **Figure 7.16**.

The results show that at an error rate of 1%, RF had the highest ranking while at 0%, RF and SVM showed insignificant differences. However, as the error rate increased, all of the algorithm prediction rankings got lower. The C5.0 model predictions showed the worst performance overall.

In general, as discussed in **Chapter 6**, the RF models displayed the highest levels of prediction accuracy rates among the ML algorithms studied. These experiments also showed that sampled instances are appropriate to estimate the best adaptive prediction threshold. These subsets were as small as 0.05% of the original evaluation dataset with a GAR close to 0.9, which is a very

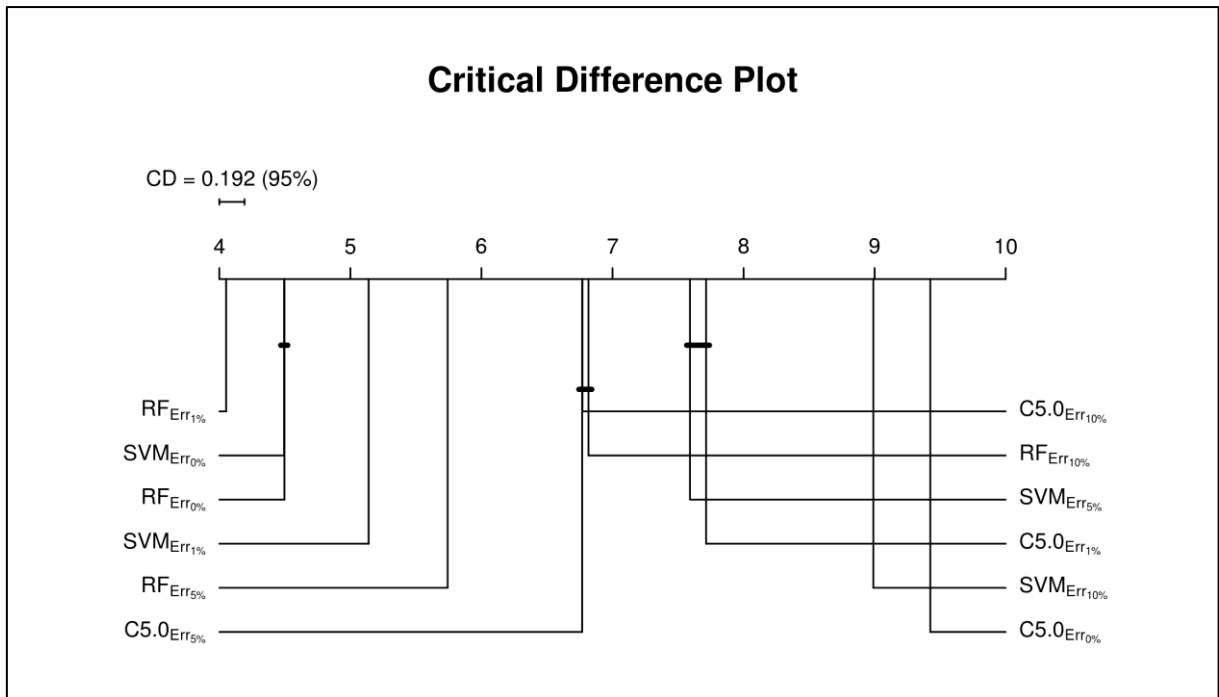


Figure 7.16: Critical Difference plots for the different ML algorithm samples under different error rates.

good ratio compared to the other algorithms. Knowing that RF predictions had the highest G-Mean Accuracy rate means the estimated sample cutoff (Thr_{Smpl}) is able to estimate up to 90% of the full model's predictive capacity if its threshold is set to Thr_{Opt} . Also, RF showed the least sensitivity to error rates in comparison to the two other algorithms. All of these findings make this algorithm the best of the three for modelling network traffic for ID tasks.

7.5 Limitations

Although the experiments outlined in this chapter investigated the practicality of selecting a 'close to optimal' adaptive threshold for evaluation (test) data based on the true label of a small subset, there were a number of limitations to them.

Firstly, the analysis used a random sampling approach to draw samples from the population being studied (the evaluation data). Other approaches could have been explored to determine the best approach for this domain such as: systematic sampling [378]; probability-proportional-to-size sampling [379, 380]; stratified sampling [288, 381, 382]; or cluster sampling [383, 384].

Secondly, only a stratified sampling method, based on a prediction range using a fixed number of bins, was investigated. Further analysis would be required to investigate other approaches which might have better techniques. For example, using some density based functions to perform a much smoother sampling would have resulted in a more representative subset of the data based on the models' predictions.

Thirdly, sampling without replacements was used. Further investigation is required to assess how sampling performs when it is applied with replacement, i.e. so that the same instances could be sampled multiple times. Such a method could be applied to the sampled subset to have better cutoff estimation within a certain confidence interval.

Fourthly, only specific percentages were analysed to examine the effect of the sample size on the identification of the '*close to optimal*' adaptive threshold. These percentage values were sufficient for this study as the datasets used in these analyses were large. However, as the size of the evaluated data gets smaller larger percentages might be needed to extract representative samples. Therefore, further investigation is needed in the future to determine the appropriate sample size given the actual data under consideration.

Finally, a thorough analysis is needed to explore the relationship between the number of unique cutoffs and the shifts in the threshold as a result of errors. The analysis outlined in this chapter showed a possible relationship between the two, but a more in-depth analysis of the causes is required. Such an analysis might result in the development of a model that could determine the correct number of cases and the best sampling strategy to apply, based on the probabilities (thresholds) returned by the prediction models. Such a model might ensure better and more representative samples based on a number of factors including, the range of the predictions, the number of unique thresholds, density distributions, model parameters and the ML algorithm.

7.6 Summary

This chapter outlines a set of empirical analyses which investigated how to determine the best optimal adaptive threshold for evaluation data. The predictions of various models, developed using different ML algorithms (C5.0, RF and SVM) with different features sets (Full, MDA, MDG, MDA_{Bal} and MDG_{Bal}) and different data balances (balanced and non-balanced), were analysed. The aim of this chapter was to identify the optimal adaptive threshold using the true label of small subsets so that such an approach could be applied to a real life setup.


The experiments outlined in this chapter analysed how the ‘*close to optimal*’ thresholds were identified, based on sampled subsets with different effects: sample size, sampling strategy and error rates. Sample thresholds were evaluated on their proximity to the optimal threshold. From these experiments it can be concluded that the larger the sample the better the threshold selected. Furthermore, it can also be concluded that as the number of bins used to sample instances from the prediction range increases, the more representative the samples will be, and hence the closer the cutoff will be to the optimal threshold.

The analysis in this chapter used the G-Mean Accuracy Ratio (GAR) to assess the quality of the selected cutoffs. The results from these experiments were assessed on this measure using the non-parametric Friedman’s test to determine whether there were any significant differences between the effects analysed.

The selected cutoffs of the samples of the RF and SVM predictions were the closest to the optimal adaptive threshold. The RF samples showed GAR values that were close to 0.9 for samples as small as 0.05%. Therefore, as the RF models have a very high G-Mean Accuracy, they will make much more accurate predictions after threshold adaptation than the use of a fixed threshold when assessing a model’s performance. These results make RF the best out of the three compared algorithms to model network traffic and detect novel intrusions.

Chapter Eight

Conclusion



From the earliest research on Intrusion Detection in 1972, researchers have employed various methods and techniques to devise systems to detect novel and unknown attacks. Various methods, such as Machine Learning (ML) and Data Mining (DM), have been used to serve this quest. However, researchers recognise the limitations of such approaches in addressing the variability in traffic patterns over time. This has led to further research in this area and a number of solutions have been proposed to address such limitations. For example, some methods have been proposed and developed to detect variability and hence adjust the parameters of the detection model in a real-time setup. Others have proposed ensemble methods to create strong ID models by aggregating and unifying multiple weak models, which can be replaced when any change in pattern is detected in a data stream domain.

However, adaptation in batch-learning methods is the area of focus for this thesis as it remains under-researched. The aim of this thesis is to demonstrate the importance of threshold tuning/adaptation of the model predictions based on the dataset being evaluated. This is due to the fact that data evolves over time, especially in dynamic environments, like network traffic, where changes occur very quickly, which renders ID models redundant faster than they would be with a fixed threshold.

To this end, in this thesis the following hypothesis has been investigated, as stated in **Chapter 1** and **Chapter 3**:

“In a binary batch-learning setup, prediction accuracy of a score-based anomaly intrusion detection model can be improved by adapting the discriminating threshold specifically for the predictions of the evaluated network traffic.”

During this research, multiple questions were addressed to piece together a holistic view of the problem. The study started by investigating the effect of adapting a discriminating threshold for every evaluation data and comparing the performance (detection accuracy) of models with a fixed threshold. The potential of threshold adaptation for a domain specific dataset (STA2018), which was specifically generated for this thesis, was then investigated under various scenarios that mimicked real life setups. Finally, this study examined the potential of threshold adaptation based on sampled validation data, where multiple variables had been used to control the sampling of the data and their effects have been assessed.

8.1 Main Findings

This section provides a summary of the main results of the experiments and analyses conducted in this thesis.

8.1.1 Importance of threshold adaptation

In a conventional binary batch-learning process, the discriminating threshold of an ID model is usually set just once using various techniques, such as a K-folds Cross-Validation or an independent validation (hold-out) data. Once that threshold has been set, it is then be used for all subsequent predictions undertaken by that model [27]. This study has highlighted both the shortcoming of such approaches and pointed out the potential of existing problems in estimating

a model's actual performance. That is because the statistical properties of the training data are likely to differ from those of both the validation data and the subsequent evaluation (test) data. As a result, in this thesis it has been suggested that threshold selection should not be based on the detection model alone, where the same threshold being used for all subsequent evaluations and be expected to provide the optimal model predictions. Instead, a discriminating threshold should be set, based on a combination of the ID model and the evaluated (test) data. Furthermore, the best threshold should be independently selected for each evaluated dataset, especially where the threshold selection is based on an optimisation of some performance criterion. As a result, this study has shown *significant statistical differences* in model performance (prediction accuracy) depending on whether a fixed or an adaptive threshold is used, with a tuned threshold improving model performance.

Three ML algorithms (C5.0, Random Forest and SVM) were compared and showed similar behaviour in relation to threshold tuning which resulted in significant improvements in the predictions of their models. However, Random Forest was the best performing algorithm with the highest detection rates and the least variability after threshold tuning.

In this research it has been concluded that the use of a fixed threshold for model predictions undermines the actual model performance. Therefore, it is recommended that the discriminating threshold should be adapted based on a representative sample of the evaluation data. However, in domains where the statistical differences (or concept drift), between the training data used to build the ID model and the evaluation (test) data is not high, threshold adaptation will incur an extra overhead, when a fixed threshold would have sufficed. Therefore, a good measure of concept drift is required, along with further investigation and research to evaluate what measures can best detect such drifts and hence help the security analysts understand when to perform threshold selection.

8.1.2 Threshold adaptation to address feature drift

The model development stage could be performed after some pre-processing phases, such as feature selection or data balancing, based on the analysis of the available data in order to improve the efficiency of the models generated. When data with different statistical properties are used, the results of such processing may not be similar. This is a common scenario in real life setups where the best features of the training data may not be the same as those of any future, unseen data. This study therefore investigated such scenarios, and evaluated multiple models under different setups.

This research concluded that for all of the ML algorithms that were compared under various model development scenarios, such as feature sets and data balancing, an adaptive threshold outperforms a fixed threshold. Essentially, threshold tuning based on the evaluated (test) data can reduce the adverse effects of such pre-processing phases (feature selection and/or data balancing) in dynamic environments.

In this thesis it has been shown that all of the ML algorithms analysed perform poorly prior to tuning. However, threshold adaptation increased the accuracy of model performance for all of the ML algorithms, with the RF algorithm being the most adaptable and showing the best overall performance. It has also been shown that *data balancing* does not produce significant improvements in model performance, except in the case of the C5.0 algorithm in those very limited instances where training data are highly skewed or imbalanced and have very few instances or records of the minority class. Similarly, different *feature sets* did not make a significant difference to the performance of the ML models, which encouraged the use of small feature sets without serious degradation in model performance (accuracy). However, with the RF algorithm, the performance of models that used feature sets selected on the basis of the Mean Decrease of Gini [for both balanced ($MDG_{Bal.}$) and imbalanced (MDG)] was statistically

lower than that of models that used other feature groups; this was due to the tendency of the Mean Decrease of Gini measure to select local features (specific to the training data).

8.1.3 Optimal threshold selection

This study has highlighted the importance of adapting the discriminating threshold for every evaluation (test) data predictions based on a representative sample of that data. In this thesis therefore it has been investigated the possibility of identifying the optimal adaptive threshold for the entire evaluation (test) data using the true labels of a sampled small subset (validation data). This investigation examined the effect of three key variables - *sample size*, *sampling strategy* and *label error rates* - on the threshold selected for the sampled validation data. The quality of the threshold selected was assessed by measuring the ratio of its performance (accuracy as measured by the Geometric Mean of Accuracy) on the entire evaluation (test) data relative to that of the optimal threshold (cutoff).

Threshold selection for the predictions of the RF and SVM algorithms, using sampled validation data, performed the best. For the RF predictions, a sample as small as 0.05% of the original evaluation (test) data was sufficient to identify a threshold with an accuracy rate of over 90% of the overall accuracy of the optimal threshold.

All of the algorithms that were compared showed similar results in terms of sample size; the bigger the sample size, the better the estimated threshold i.e. the nearer it was to the optimal threshold.

As expected, increased error rates in the sampled validation data resulted in threshold estimations that furthest from the optimal threshold. The sampled data of SVM predictions showed greater sensitivity to errors than that of RF predictions. However, the C5.0 predictions exhibited the opposite effect; as error rates increased the estimated threshold of their sampled data increased in quality, which is attributed to the inferiority of C5.0 predictions. In general, the loss of quality due to increased error rates in relation to the threshold selected, was more

evident in larger samples. However, as sample size decreased, the influence of error rates also decreased.

Stratified sampling strategies, where the prediction range is divided into multiple bins (B_{10} , B_{20} , B_{50} and B_{100}) to draw random samples, showed a more positive effect on the quality of the threshold selected than random sampling (B_1). Moreover, the larger the number of bins, the better the threshold selected. This is due to the better coverage of their prediction ranges, which results in a more representative validation sample. However, the binning effect became much more evident as samples got smaller. With sample sizes larger than 1% of the original population (data), the effect of a multi-bin sampling strategy became indistinguishable.

The relationship between error rates and the number of bins differed from one algorithm to another. The RF predictions showed a stable pattern with the larger the number of bins the closer the estimate came to the optimal threshold across all of the error rates tested. The SVM predictions showed significant differences between sampling strategies as the error rate increased. However, the C5.0 predictions were surprising; random sampling showed similar results to a large number of sampling bins, and, as error rates increased, the random sampling became significantly better at estimating the optimal threshold.

8.2 Future Work

Although the discussion of this thesis has shown the importance of threshold tuning/adaptation on the predictions of models based on the dataset under evaluation, it has also highlighted some of the limitations of this approach. Moreover, it is worth noting that, over the course of this research, more questions have been raised than initially proposed. This section therefore lists some possible directions for future studies.

8.2.1 Systematic comparison of different evaluation techniques

As outlined in the literature, K-folds Cross-Validation and hold-out are the most widely used techniques to assess the model performance of various systems and algorithms. However, the prospective technique, which generates a setup that better mimics real-life, is the most underutilised evaluation approach. Therefore, a systematic analysis is required to compare all of these techniques, to understand their true capability to assess models and to identify the best approach for setting a standard for all analyses in the IDS domain.

8.2.2 Threshold adaptation for multi-class models and other ML

This study has analysed and investigated the effect of prediction threshold tuning in binary classification setups, an approach which has been shown to significantly improved model performance. However, no analysis was conducted on multi-class problems. Therefore, one potential avenue for further investigation is extending the threshold adaptation technique to multi-class predictions.

This study also analysed three ML algorithms using their default settings. Future studies could extend this analysis to a more diverse range of ML algorithms and include other performance criteria, such as speed and resource consumption, to identify the best algorithm(s) for the network ID domain and the most adaptable to traffic variability.

8.2.3 Threshold adaptation against attack

This study did not consider the security of the IDS that implemented the tuning approach. Therefore, a further study is needed to analyse the resilience of this approach to attacks that target IDS, such as sporadic changes and evasion attacks.

8.2.4 Data stream domains

Another possible research avenue is applying the threshold tuning approach to data stream domains in order to reduce the model generation process as well as to compare its overall performance and resource utilisation with the state-of-the-art solutions in that domain. It is

expected that in such studies, the ID models will last longer as they will be phased out less frequently. This is because the ID models will maintain their high performance for longer so the updating process will not be triggered as often. However, this might require evaluating the instances in that domain in mini batches to adapt the threshold for each batch, which could slow the system response.

8.2.5 *Data pre-processing*

Once this study recognised the effect of traffic variability over time on feature importance (which can vary between training and evaluation data), it used two measures to assess feature importance and hence select the salient ones. In this thesis it has been shown that threshold tuning helped to mitigate or reduce the effect of feature drift. Therefore, further studies could analyse various feature selection and dimension reduction strategies to evaluate the best approach for the network ID domain. These studies could also thoroughly investigate which factors in threshold adaptation could mitigate the effect of feature drift on feature selection or dimension reduction methods.

Similarly, only one method, SMOTE, was used to address the imbalance effect of the training dataset on model performance. Therefore, future studies could extend the analysis to investigate how different approaches could identify the best performing technique with the least time complexity, as time is a precious requirement for network based IDS.

8.2.6 *Validation data sampling*

This study has clearly shown the advantage of threshold adaptation in improving model performance and has investigated an approach to identify the optimal adaptive threshold based on representative validation data sampled from the entire test data. The true labels of the sampled data were used to optimise the performance criteria for the selected threshold. Different *sample sizes*, *strategies* and *error rates* were also investigated to identify the optimal threshold.

A future study could use different sampling techniques to identify the ‘*close to optimal*’ threshold. It could also investigate the use of multiple thresholds based on different traffic types, given that model predictions for ICMP traffic differ from those of TCP, and the patterns of HTTP traffic differ from those of FTP or SMTP. Using a single threshold to classify diverse traffic patterns might therefore hinder the overall detection performance. As a result, a multiple thresholds setup is a line of research worth investigating further.

As this study used sampled validation data to identify a close to optimal threshold, it depended on the quality of the instances sampled. Therefore, a future study could identify the near to optimal threshold by performing iterative sampling - with replacement - using the sampled validation data. The average threshold from multiple iterations could then be assessed statistically to determine its quality.

8.2.7 Drifts measurements

The threshold adaptation process is not required when training and test datasets exhibit the same statistical properties. More studies are therefore needed to quantify the drift or difference between datasets by conducting some analysis such as Kullback–Leibler divergence test, so that such a measure can be used to assess the threshold shift for the test data. Such an approach could eliminate the need to access the true labels of the validation data, which can be a costly process.

8.2.8 Further comparisons

As the presented experiments showed that Random Forest algorithm performance was the highest between compared algorithms and the most adaptive algorithm. Also, knowing that this algorithm is a type of ensemble algorithms. Future studies could compare different ensemble type algorithms within the ID domain using the threshold adaptation approach. Further comparisons could be conducted to compare this approach with some of the state-of-the-art methods such as data stream methods and semi-supervised techniques (i.e. active learning).


8.3 Reflective/Closing Remarks

A novel approach to evaluating detection models in the network ID field has been outlined in this thesis, where the use of **prospective sampling** ensured a close resemblance to real-life setups. This study also illustrated the gap that exists between the actual performance of ID models and what is usually believed about their performance when fixed thresholds are used.

The Random Forest (RF) algorithm performed the best in detecting novel attacks and exhibited the best adaptability traits to changes in network traffic. The SVM algorithm performed second best in all the analyses conducted. Knowing that a linear version of SVM with its default setting was used throughout this thesis, a future investigation could explore the potential of this algorithm to fit the requirements of this domain. Although many studies have shown that C5.0 is comparable to algorithms such as RF and SVM, it scored the worst of all the algorithms, especially in its ability to adapt to changes in traffic.

As previous studies on IDS have adopted the conventional K-folds Cross-Validation or hold-out validation techniques in their assessment, their conclusions should be revisited as per the findings and recommendations of this thesis.

This study has provided several topical and novel contributions to IDS research as outlined in **Chapter 1**. The work has demonstrated the value of threshold adaptation in improving the prediction efficiency (accuracy) of a binary ID model in network anomaly detection. Such threshold tuning is performed using a representative small sample of the evaluated traffic.



Appendix (A) Results of Chapter 4 (First) Experiment

This appendix presents the results of the second phase experiments (conducted in **Chapter 4**) for every algorithm (C5.0, Random Forest and SVM) with different datasets (gureKDD, SEA and AGR). In these tables, each row corresponds to the model and every column represents the file that was used either in building the model or to test the model. The shaded cells present the results from where a file was used to build a prediction model using the 10-folds Cross-Validation. Each one of these cells presents two measure: the Model's optimal Threshold (**MT**); and the best G-Mean Accuracy reached, denoted by (**CA**). Every other (unshaded) cell presents three different values: the G-Mean accuracy of the model when the pre-computed cutoff (threshold) was used (**MT**) before adaptation; the new optimal threshold when it was adapted specifically to the test data file (**FT**); and the G-Mean accuracy after adaptation (**FA**).

A.1. C5.0

This section presents the performance results of C5.0 algorithm on the three different datasets (gureKDD, SEA and AGR).

	File 1	File 2	File 3	File 4	File 5	File 6	File 7
Model 1	MT: 0.0001 CA: 0.3904	MA: 0.0000 FT: 0.0000 FA: 0.0000	MA: 0.0000 FT: 0.0000 FA: 0.0000	MA: 0.0000 FT: 0.0000 FA: 0.0000	MA: 0.0000 FT: 0.0000 FA: 0.0000	MA: 0.0000 FT: 0.0000 FA: 0.0000	MA: 0.0000 FT: 0.0000 FA: 0.0000
Model 2	MA: 0.2181 FT: 0.0659 FA: 0.8125	MT: 0.3530 CA: 0.9981	MA: 0.2108 FT: 0.2140 FA: 0.9740	MA: 0.7154 FT: 0.2070 FA: 0.9248	MA: 0.2162 FT: 0.2140 FA: 0.9715	MA: 0.5786 FT: 0.2140 FA: 0.9646	MA: 0.1850 FT: 0.2140 FA: 0.9805
Model 3	MA: 0.0000 FT: 0.2007 FA: 0.8884	MA: 0.7127 FT: 0.1474 FA: 0.9150	MT: 0.5863 CA: 0.9995	MA: 0.8198 FT: 0.3735 FA: 0.8997	MA: 0.9874 FT: 0.6506 FA: 0.9956	MA: 0.9849 FT: 0.6460 FA: 0.9928	MA: 0.9981 FT: 0.3926 FA: 0.9994
Model 4	MA: 0.8727 FT: 0.0847 FA: 0.9770	MA: 0.4109 FT: 0.0856 FA: 0.8656	MA: 0.9948 FT: 0.3384 FA: 0.9949	MT: 0.3037 CA: 0.9981	MA: 0.9862 FT: 0.5781 FA: 0.9965	MA: 0.9740 FT: 0.6865 FA: 0.9944	MA: 0.9988 FT: 0.5321 FA: 0.9995
Model 5	MA: 0.8448 FT: 0.2506 FA: 0.9740	MA: 0.3315 FT: 0.0952 FA: 0.7145	MA: 0.9963 FT: 0.6314 FA: 0.9965	MA: 0.9107 FT: 0.0839 FA: 0.9453	MT: 0.5854 CA: 0.9998	MA: 0.8525 FT: 0.4144 FA: 0.9977	MA: 0.9995 FT: 0.6126 FA: 0.9995
Model 6	MA: 0.8451 FT: 0.3233 FA: 0.9997	MA: 0.9610 FT: 0.3154 FA: 0.9948	MA: 0.9986 FT: 0.6095 FA: 0.9989	MA: 0.9093 FT: 0.2791 FA: 0.9128	MA: 0.9996 FT: 0.5838 FA: 0.9997	MT: 0.5382 CA: 0.9998	MA: 0.9994 FT: 0.5608 FA: 0.9994
Model 7	MA: 0.8161 FT: 0.2371 FA: 0.9504	MA: 0.9894 FT: 0.4086 FA: 0.9903	MA: 0.8435 FT: 0.7801 FA: 0.9908	MA: 0.8454 FT: 0.6245 FA: 0.9308	MA: 0.9321 FT: 0.7427 FA: 0.9959	MA: 0.9802 FT: 0.7330 FA: 0.9939	MT: 0.4664 CA: 0.9998

Table A.1: C5.0 model's performance on gureKDD dataset with various effects (before and after threshold adaptation). MT (Model optimal Threshold); CA (Cross-validation G-Mean Accuracy); MA (Model threshold G-Mean Accuracy); FT (File optimal Threshold); FA (File threshold G-Mean Accuracy). Shaded cells are the 10-folds Cross-Validation results from the model generation phase.

	File 1	File 2	File 3	File 4	File 5	File 6
Model 1	MT: 0.4424 CA: 0.8731	MA: 0.8740 FT: 0.4584 FA: 0.8744	MA: 0.8046 FT: 0.1404 FA: 0.8517	MA: 0.8052 FT: 0.1404 FA: 0.8522	MA: 0.8361 FT: 0.1430 FA: 0.8502	MA: 0.8362 FT: 0.1404 FA: 0.8503
Model 2	MA: 0.8726 FT: 0.4338 FA: 0.8736	MT: 0.3845 CA: 0.8731	MA: 0.8086 FT: 0.0733 FA: 0.8486	MA: 0.8074 FT: 0.0736 FA: 0.8493	MA: 0.8373 FT: 0.1561 FA: 0.8471	MA: 0.8372 FT: 0.1561 FA: 0.8468
Model 3	MA: 0.8320 FT: 0.8520 FA: 0.8574	MA: 0.8319 FT: 0.8520 FA: 0.8586	MT: 0.4882 CA: 0.8898	MA: 0.8896 FT: 0.4691 FA: 0.8901	MA: 0.8592 FT: 0.5192 FA: 0.8599	MA: 0.8593 FT: 0.5181 FA: 0.8600
Model 4	MA: 0.8317 FT: 0.9259 FA: 0.8612	MA: 0.8319 FT: 0.9259 FA: 0.8617	MA: 0.8906 FT: 0.4832 FA: 0.8906	MT: 0.4827 CA: 0.8902	MA: 0.8599 FT: 0.5177 FA: 0.8603	MA: 0.8599 FT: 0.5155 FA: 0.8603
Model 5	MA: 0.8387 FT: 0.7024 FA: 0.8700	MA: 0.8394 FT: 0.7030 FA: 0.8704	MA: 0.8781 FT: 0.2805 FA: 0.8821	MA: 0.8775 FT: 0.2805 FA: 0.8821	MT: 0.2959 CA: 0.8567	MA: 0.8568 FT: 0.2948 FA: 0.8569
Model 6	MA: 0.8391 FT: 0.6915 FA: 0.8686	MA: 0.8395 FT: 0.6873 FA: 0.8691	MA: 0.8762 FT: 0.2457 FA: 0.8819	MA: 0.8759 FT: 0.2457 FA: 0.8821	MA: 0.8563 FT: 0.2773 FA: 0.8570	MT: 0.3049 CA: 0.8559

Table A.2: C5.0 model's performance on SEA dataset with various effects (before and after threshold adaptation). MT (Model optimal Threshold); CA (Cross-validation G-Mean Accuracy); MA (Model threshold G-Mean Accuracy); FT (File optimal Threshold); FA (File threshold G-Mean Accuracy). Shaded cells are the 10-folds Cross-Validation results from the model generation phase.

	File 1	File 2	File 3	File 4	File 5	File 6
Model 1	MT: 0.7225 CA: 0.9449	MA: 0.9443 FT: 0.7171 FA: 0.9445	MA: 0.4844 FT: 0.9374 FA: 0.4932	MA: 0.4850 FT: 0.9383 FA: 0.4930	MA: 0.6873 FT: 0.8237 FA: 0.6888	MA: 0.6873 FT: 0.8212 FA: 0.6885
Model 2	MA: 0.9447 FT: 0.7078 FA: 0.9448	MT: 0.7207 CA: 0.9448	MA: 0.4835 FT: 0.9425 FA: 0.4938	MA: 0.4829 FT: 0.9426 FA: 0.4934	MA: 0.6871 FT: 0.8293 FA: 0.6882	MA: 0.6867 FT: 0.8141 FA: 0.6882
Model 3	MA: 0.4925 FT: 0.2948 FA: 0.4932	MA: 0.4925 FT: 0.2880 FA: 0.4929	MT: 0.3838 CA: 0.9341	MA: 0.9341 FT: 0.3843 FA: 0.9341	MA: 0.6968 FT: 0.2862 FA: 0.6984	MA: 0.6976 FT: 0.2850 FA: 0.6990
Model 4	MA: 0.4907 FT: 0.3032 FA: 0.4916	MA: 0.4900 FT: 0.2537 FA: 0.4911	MA: 0.9328 FT: 0.3757 FA: 0.9334	MT: 0.3917 CA: 0.9339	MA: 0.6956 FT: 0.2813 FA: 0.6977	MA: 0.6964 FT: 0.2628 FA: 0.6985
Model 5	MA: 0.7114 FT: 0.3538 FA: 0.7492	MA: 0.7114 FT: 0.3588 FA: 0.7484	MA: 0.7382 FT: 0.6698 FA: 0.7623	MA: 0.7386 FT: 0.6669 FA: 0.7624	MT: 0.5418 CA: 0.7059	MA: 0.7079 FT: 0.5505 FA: 0.7081
Model 6	MA: 0.7147 FT: 0.3788 FA: 0.7463	MA: 0.7140 FT: 0.3790 FA: 0.7459	MA: 0.7360 FT: 0.6740 FA: 0.7626	MA: 0.7376 FT: 0.6807 FA: 0.7628	MA: 0.7082 FT: 0.5236 FA: 0.7085	MT: 0.5259 CA: 0.7101

Table A.3: *CS.0* model's performance on *AGR* dataset with various effects (before and after threshold adaptation). MT (Model optimal Threshold); CA (Cross-validation G-Mean Accuracy); MA (Model threshold G-Mean Accuracy); FT (File optimal Threshold); FA (File threshold G-Mean Accuracy). Shaded cells are the 10-folds Cross-Validation results from the model generation phase.

A.2. Random Forest

This section presents the performance results of the Random Forest (RF) algorithm on the three different datasets (gureKDD, SEA and AGR).

	File 1	File 2	File 3	File 4	File 5	File 6	File 7
Model 1	MT: 0.0004 CA: 0.9987	MA: 0.9752 FT: 0.0020 FA: 0.9777	MA: 0.8538 FT: 0.0247 FA: 0.9914	MA: 0.7948 FT: 0.0197 FA: 0.9423	MA: 0.6733 FT: 0.0790 FA: 0.9937	MA: 0.7410 FT: 0.0765 FA: 0.9929	MA: 0.9673 FT: 0.0219 FA: 0.9952
Model 2	MA: 0.3085 FT: 0.0021 FA: 0.9930	MT: 0.0682 CA: 0.9984	MA: 0.9807 FT: 0.0620 FA: 0.9851	MA: 0.9103 FT: 0.0259 FA: 0.9359	MA: 0.9657 FT: 0.0634 FA: 0.9699	MA: 0.9531 FT: 0.0651 FA: 0.9563	MA: 0.9859 FT: 0.0491 FA: 0.9963
Model 3	MA: 0.2182 FT: 0.0702 FA: 0.9205	MA: 0.6430 FT: 0.0728 FA: 0.9902	MT: 0.7623 CA: 0.9996	MA: 0.5304 FT: 0.2848 FA: 0.9324	MA: 0.9898 FT: 0.5749 FA: 0.9951	MA: 0.8262 FT: 0.4472 FA: 0.9930	MA: 0.9815 FT: 0.3997 FA: 0.9994
Model 4	MA: 0.8448 FT: 0.0671 FA: 0.9970	MA: 0.7060 FT: 0.0479 FA: 0.9894	MA: 0.9953 FT: 0.4800 FA: 0.9966	MT: 0.2299 CA: 0.9983	MA: 0.9862 FT: 0.4524 FA: 0.9990	MA: 0.9747 FT: 0.6083 FA: 0.9947	MA: 0.9987 FT: 0.4758 FA: 0.9995
Model 5	MA: 0.8165 FT: 0.1205 FA: 0.9736	MA: 0.6326 FT: 0.0100 FA: 0.8836	MA: 0.9968 FT: 0.5226 FA: 0.9969	MA: 0.9311 FT: 0.0468 FA: 0.9418	MT: 0.6004 CA: 0.9999	MA: 0.9980 FT: 0.4297 FA: 0.9981	MA: 0.9996 FT: 0.5539 FA: 0.9996
Model 6	MA: 0.8863 FT: 0.2246 FA: 0.9981	MA: 0.9542 FT: 0.1032 FA: 0.9965	MA: 0.9989 FT: 0.4994 FA: 0.9991	MA: 0.9082 FT: 0.0270 FA: 0.9486	MA: 0.9998 FT: 0.5236 FA: 0.9998	MT: 0.6015 CA: 0.9999	MA: 0.9994 FT: 0.3762 FA: 0.9996
Model 7	MA: 0.8448 FT: 0.0667 FA: 0.9754	MA: 0.9841 FT: 0.1118 FA: 0.9908	MA: 0.9884 FT: 0.4181 FA: 0.9986	MA: 0.9300 FT: 0.3612 FA: 0.9352	MA: 0.9914 FT: 0.4174 FA: 0.9970	MA: 0.9961 FT: 0.3643 FA: 0.9974	MT: 0.3046 CA: 0.9999

Table A.4: *Random Forest (RF)* model's performance on *gureKDD* dataset with various effects (before and after threshold adaptation). MT (Model optimal Threshold); CA (Cross-validation G-Mean Accuracy); MA (Model threshold G-Mean Accuracy); FT (File optimal Threshold); FA (File threshold G-Mean Accuracy). Shaded cells are the 10-folds Cross-Validation results from the model generation phase.

	File 1	File 2	File 3	File 4	File 5	File 6
Model 1	MT: 0.4249 CA: 0.8750	MA: 0.8758 FT: 0.4352 FA: 0.8758	MA: 0.8696 FT: 0.1905 FA: 0.8764	MA: 0.8027 FT: 0.2857 FA: 0.8053	MA: 0.8358 FT: 0.3469 FA: 0.8364	MA: 0.8358 FT: 0.3492 FA: 0.8364
Model 2	MA: 0.8752 FT: 0.4455 FA: 0.8752	MT: 0.4286 CA: 0.8757	MA: 0.8026 FT: 0.2770 FA: 0.8053	MA: 0.8680 FT: 0.1931 FA: 0.8759	MA: 0.8357 FT: 0.3565 FA: 0.8363	MA: 0.8357 FT: 0.3501 FA: 0.8363
Model 3	MA: 0.8536 FT: 0.3614 FA: 0.9113	MA: 0.8323 FT: 0.6493 FA: 0.8338	MT: 0.4942 CA: 0.8920	MA: 0.8924 FT: 0.4518 FA: 0.8926	MA: 0.8609 FT: 0.5025 FA: 0.8610	MA: 0.8604 FT: 0.5672 FA: 0.8607
Model 4	MA: 0.8319 FT: 0.6519 FA: 0.8333	MA: 0.8636 FT: 0.3710 FA: 0.9113	MA: 0.8921 FT: 0.5081 FA: 0.8921	MT: 0.4795 CA: 0.8925	MA: 0.8609 FT: 0.5323 FA: 0.8612	MA: 0.8606 FT: 0.5489 FA: 0.8609
Model 5	MA: 0.8389 FT: 0.6723 FA: 0.8685	MA: 0.8393 FT: 0.6730 FA: 0.8695	MA: 0.8782 FT: 0.2943 FA: 0.8832	MA: 0.8786 FT: 0.2998 FA: 0.8839	MT: 0.3635 CA: 0.8576	MA: 0.8572 FT: 0.3373 FA: 0.8576
Model 6	MA: 0.8369 FT: 0.6597 FA: 0.8691	MA: 0.8368 FT: 0.6596 FA: 0.8695	MA: 0.8806 FT: 0.3000 FA: 0.8829	MA: 0.8815 FT: 0.3111 FA: 0.8835	MA: 0.8579 FT: 0.3383 FA: 0.8579	MT: 0.3391 CA: 0.8574

Table A.5: *Random Forest (RF)* model's performance on *SEA* dataset with various effects (before and after threshold adaptation). MT (Model optimal Threshold); CA (Cross-validation G-Mean Accuracy); MA (Model threshold G-Mean Accuracy); FT (File optimal Threshold); FA (File threshold G-Mean Accuracy). Shaded cells are the 10-folds Cross-Validation results from the model generation phase.

	File 1	File 2	File 3	File 4	File 5	File 6
Model 1	MT: 0.6669 CA: 0.9483	MA: 0.9482 FT: 0.6423 FA: 0.9484	MA: 0.4774 FT: 0.9975 FA: 0.5042	MA: 0.4812 FT: 0.9954 FA: 0.5040	MA: 0.6869 FT: 0.8261 FA: 0.6895	MA: 0.6871 FT: 0.8361 FA: 0.6893
Model 2	MA: 0.9488 FT: 0.6482 FA: 0.9490	MT: 0.6291 CA: 0.9486	MA: 0.4788 FT: 0.9951 FA: 0.5059	MA: 0.4762 FT: 0.9972 FA: 0.5054	MA: 0.6858 FT: 0.8200 FA: 0.6894	MA: 0.6856 FT: 0.8227 FA: 0.6895
Model 3	MA: 0.4900 FT: 0.2258 FA: 0.4939	MA: 0.4933 FT: 0.3376 FA: 0.4940	MT: 0.4067 CA: 0.9387	MA: 0.9387 FT: 0.3566 FA: 0.9395	MA: 0.6989 FT: 0.3066 FA: 0.7007	MA: 0.6995 FT: 0.3278 FA: 0.7019
Model 4	MA: 0.4928 FT: 0.2955 FA: 0.4939	MA: 0.4902 FT: 0.2093 FA: 0.4942	MA: 0.9390 FT: 0.3897 FA: 0.9391	MT: 0.4040 CA: 0.9398	MA: 0.6995 FT: 0.3192 FA: 0.7011	MA: 0.6997 FT: 0.3007 FA: 0.7016
Model 5	MA: 0.7208 FT: 0.4309 FA: 0.7620	MA: 0.7212 FT: 0.4309 FA: 0.7620	MA: 0.7401 FT: 0.6208 FA: 0.7785	MA: 0.7402 FT: 0.6201 FA: 0.7779	MT: 0.5182 CA: 0.7127	MA: 0.7144 FT: 0.5122 FA: 0.7149
Model 6	MA: 0.7248 FT: 0.4229 FA: 0.7607	MA: 0.7243 FT: 0.4310 FA: 0.7608	MA: 0.7351 FT: 0.6186 FA: 0.7788	MA: 0.7367 FT: 0.6119 FA: 0.7805	MA: 0.7139 FT: 0.5125 FA: 0.7140	MT: 0.5133 CA: 0.7129

Table A.6: *Random Forest (RF)* model's performance on *AGR* dataset with various effects (before and after threshold adaptation). MT (Model optimal Threshold); CA (Cross-validation G-Mean Accuracy); MA (Model threshold G-Mean Accuracy); FT (File optimal Threshold); FA (File threshold G-Mean Accuracy). Shaded cells are the 10-folds Cross-Validation results from the model generation phase.

A.3. Support Vector Machine (SVM)

This section presents the performance results of the SVM algorithm on the three different datasets (gureKDD, SEA and AGR).

	File 1	File 2	File 3	File 4	File 5	File 6	File 7
Model 1	MT: 0.9495 CA: 0.9250	MA: 0.6471 FT: 0.5556 FA: 0.6545	MA: 0.8171 FT: 0.9969 FA: 0.9727	MA: 0.2401 FT: 0.7165 FA: 0.3504	MA: 0.8974 FT: 0.9968 FA: 0.9695	MA: 0.8250 FT: 0.8581 FA: 0.9123	MA: 0.9665 FT: 0.9968 FA: 0.9715
Model 2	MA: 0.0000 FT: 0.9999 FA: 0.8253	MT: 1.0000 CA: 0.9869	MA: 0.1701 FT: 0.9520 FA: 0.5024	MA: 0.3076 FT: 0.9414 FA: 0.4525	MA: 0.1176 FT: 0.9219 FA: 0.5163	MA: 0.3457 FT: 0.9339 FA: 0.5285	MA: 0.1116 FT: 0.9425 FA: 0.6131
Model 3	MA: 0.8092 FT: 0.0070 FA: 0.8303	MA: 0.7206 FT: 0.0009 FA: 0.9022	MT: 0.0134 CA: 0.9977	MA: 0.7544 FT: 0.0004 FA: 0.9028	MA: 0.9699 FT: 0.0647 FA: 0.9793	MA: 0.9583 FT: 0.0272 FA: 0.9636	MA: 0.9791 FT: 0.0571 FA: 0.9878
Model 4	MA: 0.9195 FT: 0.9454 FA: 0.9196	MA: 0.2794 FT: 0.2162 FA: 0.6683	MA: 0.9678 FT: 0.9997 FA: 0.9941	MT: 0.9392 CA: 0.9591	MA: 0.9766 FT: 0.9990 FA: 0.9958	MA: 0.9783 FT: 0.9703 FA: 0.9840	MA: 0.9867 FT: 0.9989 FA: 0.9986
Model 5	MA: 0.8724 FT: 0.0000 FA: 0.9757	MA: 0.2233 FT: 0.0000 FA: 0.6778	MA: 0.9865 FT: 0.0065 FA: 0.9922	MA: 0.9172 FT: 0.0023 FA: 0.9339	MT: 0.0296 CA: 0.9992	MA: 0.8503 FT: 0.0068 FA: 0.8507	MA: 0.9983 FT: 0.0956 FA: 0.9985
Model 6	MA: 0.8443 FT: 0.0065 FA: 0.9531	MA: 0.2804 FT: 0.0077 FA: 0.6929	MA: 0.9894 FT: 0.8938 FA: 0.9907	MA: 0.9145 FT: 0.2803 FA: 0.9270	MA: 0.9976 FT: 0.7580 FA: 0.9979	MT: 0.7049 CA: 0.9970	MA: 0.9986 FT: 0.7158 FA: 0.9986
Model 7	MA: 0.8165 FT: 0.0025 FA: 0.8518	MA: 0.3163 FT: 0.0028 FA: 0.8853	MA: 0.9944 FT: 0.6753 FA: 0.9944	MA: 0.9107 FT: 0.0898 FA: 0.9366	MA: 0.9960 FT: 0.8415 FA: 0.9962	MA: 0.8476 FT: 0.0317 FA: 0.9434	MT: 0.7043 CA: 0.9994

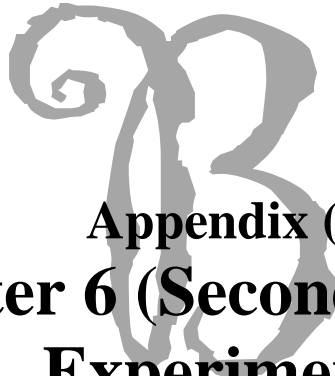
Table A.7: SVM model's performance on gureKDD dataset with various effects (before and after threshold adaptation). MT (Model optimal Threshold); CA (Cross-validation G-Mean Accuracy); MA (Model threshold G-Mean Accuracy); FT (File optimal Threshold); FA (File threshold G-Mean Accuracy). Shaded cells are the 10-folds Cross-Validation results from the model generation phase.

	File 1	File 2	File 3	File 4	File 5	File 6
Model 1	MT: 0.4995 CA: 0.8763	MA: 0.8771 FT: 0.4999 FA: 0.8771	MA: 0.8018 FT: 0.3778 FA: 0.8936	MA: 0.8016 FT: 0.3773 FA: 0.8941	MA: 0.8358 FT: 0.3778 FA: 0.8617	MA: 0.8360 FT: 0.3785 FA: 0.8615
Model 2	MA: 0.8759 FT: 0.4949 FA: 0.8760	MT: 0.4951 CA: 0.8765	MA: 0.8021 FT: 0.3734 FA: 0.8928	MA: 0.8018 FT: 0.3731 FA: 0.8933	MA: 0.8356 FT: 0.3766 FA: 0.8613	MA: 0.8359 FT: 0.3764 FA: 0.8610
Model 3	MA: 0.8319 FT: 0.6300 FA: 0.8763	MA: 0.8320 FT: 0.6304 FA: 0.8770	MT: 0.4994 CA: 0.8933	MA: 0.8939 FT: 0.4997 FA: 0.8940	MA: 0.8615 FT: 0.5010 FA: 0.8617	MA: 0.8613 FT: 0.5005 FA: 0.8615
Model 4	MA: 0.8319 FT: 0.6263 FA: 0.8763	MA: 0.8321 FT: 0.6264 FA: 0.8769	MA: 0.8933 FT: 0.4964 FA: 0.8933	MT: 0.4964 CA: 0.8938	MA: 0.8615 FT: 0.4977 FA: 0.8616	MA: 0.8612 FT: 0.4972 FA: 0.8613
Model 5	MA: 0.8325 FT: 0.5602 FA: 0.8759	MA: 0.8326 FT: 0.5598 FA: 0.8765	MA: 0.8924 FT: 0.4337 FA: 0.8928	MA: 0.8929 FT: 0.4342 FA: 0.8932	MT: 0.4358 CA: 0.8614	MA: 0.8610 FT: 0.4361 FA: 0.8611
Model 6	MA: 0.8331 FT: 0.5572 FA: 0.8756	MA: 0.8332 FT: 0.5565 FA: 0.8760	MA: 0.8914 FT: 0.4311 FA: 0.8923	MA: 0.8918 FT: 0.4311 FA: 0.8927	MA: 0.8612 FT: 0.4346 FA: 0.8612	MT: 0.4351 CA: 0.8609

Table A.8: SVM model's performance on SEA dataset with various effects (before and after threshold adaptation). MT (Model optimal Threshold); CA (Cross-validation G-Mean Accuracy); MA (Model threshold G-Mean Accuracy); FT (File optimal Threshold); FA (File threshold G-Mean Accuracy). Shaded cells are the 10-folds Cross-Validation results from the model generation phase.

	File 1	File 2	File 3	File 4	File 5	File 6
Model 1	MT: 0.6778 CA: 0.5529	MA: 0.5614 FT: 0.6778 FA: 0.5615	MA: 0.4695 FT: 0.6720 FA: 0.5106	MA: 0.4676 FT: 0.6721 FA: 0.5079	MA: 0.5148 FT: 0.6759 FA: 0.5211	MA: 0.5112 FT: 0.6753 FA: 0.5178
Model 2	MA: 0.5494 FT: 0.6769 FA: 0.5498	MT: 0.6767 CA: 0.5479	MA: 0.4829 FT: 0.6723 FA: 0.5045	MA: 0.4813 FT: 0.6720 FA: 0.5032	MA: 0.5148 FT: 0.6750 FA: 0.5174	MA: 0.5125 FT: 0.6749 FA: 0.5161
Model 3	MA: 0.4879 FT: 0.3848 FA: 0.4995	MA: 0.4877 FT: 0.3871 FA: 0.5004	MT: 0.3492 CA: 0.6440	MA: 0.6460 FT: 0.3463 FA: 0.6462	MA: 0.5656 FT: 0.3510 FA: 0.5659	MA: 0.5676 FT: 0.3553 FA: 0.5685
Model 4	MA: 0.4862 FT: 0.3895 FA: 0.4991	MA: 0.4861 FT: 0.3939 FA: 0.5005	MA: 0.6450 FT: 0.3487 FA: 0.6453	MT: 0.3476 CA: 0.6467	MA: 0.5652 FT: 0.3541 FA: 0.5664	MA: 0.5673 FT: 0.3553 FA: 0.5688
Model 5	MA: 0.4867 FT: 0.5308 FA: 0.4990	MA: 0.4862 FT: 0.5302 FA: 0.5003	MA: 0.6338 FT: 0.5078 FA: 0.6348	MA: 0.6352 FT: 0.5065 FA: 0.6365	MT: 0.5114 CA: 0.5598	MA: 0.5620 FT: 0.5122 FA: 0.5623
Model 6	MA: 0.4892 FT: 0.5345 FA: 0.4996	MA: 0.4889 FT: 0.5345 FA: 0.5006	MA: 0.6357 FT: 0.5118 FA: 0.6374	MA: 0.6362 FT: 0.5104 FA: 0.6384	MA: 0.5615 FT: 0.5140 FA: 0.5617	MT: 0.5159 CA: 0.5632

Table A.9: *SVM* model's performance on *AGR* dataset with various effects (before and after threshold adaptation). MT (Model optimal Threshold); CA (Cross-validation G-Mean Accuracy); MA (Model threshold G-Mean Accuracy); FT (File optimal Threshold); FA (File threshold G-Mean Accuracy). Shaded cells are the 10-folds Cross-Validation results from the model generation phase.



Appendix (B) Results of Chapter 6 (Second) Experiment

This appendix lists the results of the second set of experiments performed on the STA2018 dataset and discussed in **Chapter 6**.

B.1. Selected Features

This section lists the results of the feature selection stage discussed in **Section 6.2** (see **Chapter 6**). The following tables show the selected features for every simulation day in the **STA2018** dataset for every feature importance measure.

B.1.1. Day 2 (12/Jun)

Features Set	Number of Features	Features Indexes
MDA	131	1, 4, 7, 12, 15, 16, 17, 20, 25, 26, 27, 28, 29, 30, 36, 49, 58, 59, 60, 62, 63, 64, 72, 73, 74, 75, 76, 77, 79, 89, 90, 91, 92, 94, 96, 98, 99, 100, 101, 105, 106, 107, 109, 110, 111, 112, 114, 116, 117, 118, 121, 122, 125, 126, 127, 134, 137, 138, 141, 142, 144, 149, 150, 151, 154, 155, 158, 168, 177, 178, 179, 180, 182, 196, 199, 201, 209, 214, 220, 222, 226, 232, 233, 235, 237, 239, 241, 264, 265, 268, 269, 273, 275, 277, 279, 281, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 297, 300, 304, 306, 309, 312, 321, 324, 329, 340, 345, 348, 350, 367, 372, 381, 383, 401, 437, 442, 465, 476, 478, 482, 490
MDG	124	4, 7, 12, 15, 16, 17, 18, 19, 22, 25, 26, 27, 30, 35, 36, 37, 38, 40, 44, 45, 49, 57, 58, 59, 60, 62, 63, 64, 72, 73, 74, 75, 76, 77, 79, 88, 89, 90, 91, 92, 94, 97, 98, 99, 100, 101, 105, 106, 109, 110, 111, 112, 114, 116, 117, 118, 121, 122, 125, 126, 127, 134, 137, 138, 139, 141, 142, 143, 147, 151, 153, 156, 158, 162, 177, 178, 179, 180, 181, 182, 196, 198, 199, 203, 204, 209, 222, 226, 231, 233, 237, 239, 241, 251, 261, 263, 264, 268, 269, 273, 275, 277, 286, 287, 288, 289, 290, 292, 293, 294, 295, 300, 306, 329, 340, 345, 350, 380, 381, 383, 476, 478, 506, 530
MDA _{Bal.}	166	4, 7, 8, 12, 17, 18, 20, 22, 26, 27, 30, 35, 36, 37, 42, 44, 45, 47, 49, 51, 52, 53, 59, 60, 61, 62, 63, 64, 73, 74, 75, 76, 77, 78, 79, 89, 90, 91, 94, 96, 97, 98, 99, 101, 102, 104, 106, 108, 109, 110, 111, 112, 114, 116, 117, 118, 119, 121, 122, 124, 127, 136, 137, 139, 141, 143, 144, 150, 151, 154, 155, 156, 157, 158, 162, 163, 166, 168, 176, 182, 187, 196, 199, 200, 205, 209, 212, 214, 216, 220, 222, 226, 231, 232, 233, 234, 237, 238, 239, 240, 241, 247, 254, 263, 264, 265, 268, 269, 272, 273, 275, 277, 279, 281, 283, 286, 287, 288, 289, 290, 292, 293, 294, 295, 297, 298, 300, 302, 304, 306, 312, 316, 318, 319, 321, 322, 324, 327, 329, 333, 340, 341, 345, 351, 360, 382, 383, 399, 400, 417, 437, 462, 469, 476, 478, 482, 490, 491, 493, 519, 521, 530, 531, 538, 542, 546
MDG _{Bal.}	119	4, 7, 8, 12, 29, 35, 36, 37, 42, 45, 47, 49, 51, 53, 57, 58, 59, 61, 62, 63, 64, 70, 72, 76, 77, 78, 79, 88, 89, 90, 91, 92, 94, 96, 99, 101, 102, 104, 106, 108, 109, 110, 111, 112, 116, 122, 124, 135, 140, 148, 151, 152, 155, 157, 168, 198, 199, 200, 204, 209, 212, 214, 226, 227, 232, 234, 235, 236, 237, 239, 241, 250, 255, 263, 269, 272, 273, 275, 277, 279, 281, 283, 286, 288, 289, 290, 292, 293, 294, 295, 296, 297, 310, 311, 319, 327, 329, 341, 345, 347, 349, 351, 379, 382, 432, 462, 470, 476, 478, 480, 487, 491, 492, 507, 519, 523, 524, 534, 538

B.1.2. Day 3 (13/Jun)

Features Set	Number of Features	Features Indexes
MDA	519	1, 4, 7, 8, 12, 15, 16, 17, 18, 19, 20, 21, 22, 25, 26, 27, 28, 29, 30, 35, 36, 37, 38, 39, 40, 41, 42, 44, 45, 46, 47, 48, 49, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180, 181, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 219, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 253, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 301, 302, 303, 304, 305, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 387, 388, 389, 390, 391, 392, 393, 394, 395, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 409, 410, 411, 412, 413, 414, 415, 416, 417, 418, 419, 420, 421, 422, 423, 424, 425, 426, 427, 428, 429, 430, 431, 432, 433, 435, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 449, 450, 451, 453, 454, 455, 458, 459, 461, 462, 463, 464, 465, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549
MDG	27	7, 202, 230, 284, 312, 327, 335, 340, 344, 348, 390, 403, 406, 408, 413, 431, 445, 447, 453, 459, 485, 513, 516, 518, 528, 545, 549
MDA_{Bal.}	508	1, 4, 7, 8, 12, 15, 16, 17, 18, 19, 20, 21, 22, 25, 26, 27, 28, 29, 30, 35, 36, 37, 38, 39, 40, 41, 42, 44, 45, 46, 47, 48, 49, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167, 168, 169, 171, 172, 173, 174, 175, 177, 178, 179, 180, 181, 182, 183, 184, 185, 187, 188, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 220, 221, 222, 223, 224, 225, 226, 227, 228, 229, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 325, 326, 327, 328, 329, 330, 331, 332, 333, 334, 335, 336, 337, 338, 339, 340, 341, 342, 343, 344, 345, 346, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 364, 365, 366, 367, 368, 369, 370, 371, 372, 374, 375, 376, 378, 379, 382, 383, 385, 386, 387, 389, 390, 391, 392, 393, 394, 395, 396, 397, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 410, 411, 412, 413, 414, 416, 417, 418, 421, 423, 424, 425, 426, 428, 430, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 446, 447, 448, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 463, 464, 466, 467, 468, 469, 470, 471, 472, 473, 474, 475, 476, 477, 478, 479, 480, 481, 482, 483, 484, 485, 486, 487, 488, 489, 490, 492, 493, 494, 495, 496, 498, 500, 501, 502, 503, 504, 505, 506, 507, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 526, 527, 528, 529, 530, 531, 532, 533, 534, 535, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 546, 547, 548, 549
MDG_{Bal.}	138	1, 4, 7, 8, 12, 15, 16, 17, 22, 26, 27, 59, 62, 63, 64, 65, 66, 67, 73, 74, 75, 77, 78, 79, 89, 91, 94, 97, 98, 99, 112, 127, 137, 138, 146, 147, 152, 153, 154, 155, 158, 164, 166, 168, 178, 180, 182, 198, 200, 202, 205, 209, 210, 212, 216, 230, 231, 232, 233, 234, 235, 236, 239, 240, 241, 243, 245, 247, 249, 250, 251, 252, 258, 260, 263, 264, 267, 269, 271, 277, 281, 282, 284, 296, 312, 316, 321, 322, 323, 324, 327, 333, 334, 335, 337, 340, 344, 347, 348, 349, 350, 353, 355, 356, 368, 372, 383, 403, 406, 408, 413, 417, 428, 431, 440, 441, 443, 447, 453, 459, 470, 485, 487, 488, 489, 491, 511, 513, 516, 518, 520, 528, 536, 537, 541, 545, 548, 549

B.1.3. Day 4 (14/Jun)

Features Set	Number of Features	Features Indexes
MDA	365	1, 4, 7, 8, 12, 15, 16, 17, 18, 19, 20, 22, 25, 26, 27, 28, 29, 30, 35, 36, 38, 40, 44, 45, 46, 47, 48, 49, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 72, 73, 74, 75, 76, 77, 78, 79, 87, 88, 89, 90, 91, 92, 94, 97, 98, 99, 100, 101, 102, 103, 104, 105, 108, 109, 110, 111, 112, 116, 117, 118, 119, 121, 122, 123, 124, 125, 126, 127, 131, 137, 138, 139, 140, 141, 142, 143, 144, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 162, 163, 164, 166, 168, 172, 176, 177, 178, 180, 182, 184, 185, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 216, 218, 220, 222, 224, 226, 227, 228, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 254, 257, 258, 259, 260, 262, 263, 264, 265, 267, 268, 269, 271, 273, 275, 276, 277, 279, 281, 282, 283, 284, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 302, 304, 305, 306, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 320, 321, 322, 323, 324, 327, 329, 333, 334, 335, 336, 337, 340, 341, 344, 345, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 366, 367, 368, 370, 372, 378, 379, 380, 381, 382, 383, 384, 386, 388, 396, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 412, 413, 417, 421, 425, 428, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 444, 445, 447, 449, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 462, 464, 465, 469, 470, 474, 476, 478, 480, 482, 484, 485, 487, 488, 489, 490, 491, 492, 493, 494, 496, 497, 498, 506, 508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 522, 523, 524, 525, 528, 530, 534, 535, 536, 537, 538, 541, 542, 545, 546, 548, 549
MDG	11	163, 277, 352, 362, 366, 367, 370, 372, 464, 469, 474
MDA_{Bal.}	379	1, 4, 7, 8, 12, 15, 16, 17, 18, 19, 20, 22, 25, 26, 27, 28, 35, 37, 38, 40, 42, 44, 45, 46, 47, 48, 49, 51, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 70, 72, 73, 74, 75, 77, 78, 79, 87, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 114, 117, 118, 119, 121, 122, 123, 124, 126, 127, 128, 130, 131, 132, 137, 138, 139, 140, 141, 142, 143, 144, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 160, 162, 163, 164, 166, 168, 170, 171, 174, 176, 178, 179, 180, 181, 182, 184, 194, 195, 196, 197, 198, 199, 200, 201, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 214, 216, 218, 220, 222, 224, 226, 227, 228, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 243, 244, 245, 246, 247, 248, 249, 250, 251, 252, 253, 255, 256, 257, 258, 259, 260, 261, 262, 263, 264, 265, 267, 268, 269, 271, 273, 274, 275, 276, 277, 279, 281, 282, 283, 284, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 299, 300, 302, 303, 304, 305, 307, 308, 309, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 321, 322, 323, 324, 327, 329, 332, 333, 334, 335, 336, 337, 338, 340, 341, 344, 345, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 366, 367, 368, 370, 372, 373, 375, 376, 377, 378, 379, 380, 381, 382, 383, 386, 387, 396, 397, 398, 399, 400, 401, 402, 403, 404, 405, 406, 407, 408, 412, 413, 417, 421, 423, 425, 428, 431, 432, 433, 434, 435, 436, 437, 439, 440, 441, 442, 443, 444, 445, 446, 450, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 464, 466, 469, 474, 476, 477, 478, 480, 484, 485, 487, 489, 490, 491, 492, 493, 495, 497, 503, 506, 507, 508, 509, 510, 512, 513, 514, 515, 516, 517, 518, 519, 520, 522, 523, 524, 525, 528, 530, 534, 535, 536, 537, 538, 539, 541, 542, 545, 546, 548, 549
MDG_{Bal.}	118	1, 4, 7, 8, 12, 15, 16, 17, 18, 22, 25, 26, 27, 28, 37, 47, 58, 59, 60, 65, 66, 67, 72, 73, 74, 75, 89, 90, 91, 97, 98, 99, 103, 104, 105, 106, 112, 114, 137, 140, 146, 147, 149, 150, 152, 153, 154, 155, 157, 158, 162, 163, 164, 166, 168, 178, 179, 198, 205, 216, 227, 235, 242, 251, 256, 259, 264, 268, 269, 273, 277, 283, 286, 287, 288, 289, 291, 294, 297, 298, 300, 308, 316, 319, 322, 323, 324, 327, 334, 337, 340, 350, 352, 356, 357, 362, 366, 370, 372, 379, 397, 398, 406, 412, 417, 435, 436, 437, 452, 457, 460, 464, 469, 474, 487, 528, 541, 549

B.1.4. Day 5 (15/Jun)

Features Set	Number of Features	Features Indexes
MDA	369	1, 4, 7, 8, 12, 15, 16, 17, 18, 19, 20, 22, 25, 26, 27, 28, 29, 30, 35, 36, 37, 38, 39, 40, 44, 45, 46, 47, 48, 49, 57, 58, 59, 60, 62, 63, 64, 65, 72, 73, 74, 75, 76, 77, 89, 90, 91, 92, 94, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 108, 109, 110, 111, 112, 113, 114, 116, 117, 118, 119, 121, 122, 123, 124, 125, 126, 127, 131, 132, 135, 137, 138, 139, 140, 141, 142, 143, 144, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 162, 163, 164, 166, 168, 170, 172, 174, 176, 177, 178, 179, 180, 181, 182, 183, 185, 186, 187, 188, 195, 196, 198, 199, 200, 201, 202, 204, 205, 206, 208, 209, 210, 211, 212, 214, 216, 218, 220, 222, 224, 226, 227, 228, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 243, 245, 247, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 263, 264, 265, 267, 268, 269, 271, 273, 275, 277, 279, 281, 282, 283, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 300, 302, 304, 305, 306, 308, 309, 310, 311, 313, 314, 315, 316, 317, 318, 319, 321, 322, 323, 324, 327, 329, 333, 334, 335, 336, 337, 338, 340, 341, 345, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 366, 367, 368, 370, 372, 373, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 386, 388, 389, 396, 397, 399, 400, 401, 402, 403, 404, 405, 406, 407, 410, 412, 413, 415, 417, 419, 421, 423, 425, 427, 428, 429, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 451, 452, 453, 454, 455, 456, 457, 458, 459, 460, 461, 462, 464, 465, 468, 469, 470, 472, 474, 476, 480, 482, 484, 486, 487, 488, 489, 490, 492, 493, 494, 495, 496, 497, 498, 506, 507, 509, 510, 511, 512, 514, 515, 516, 517, 518, 519, 520, 522, 523, 524, 525, 528, 530, 534, 535, 537, 538, 539, 541, 542, 546, 548, 549
MDG	114	1, 4, 8, 12, 15, 17, 19, 20, 22, 25, 26, 27, 29, 30, 37, 38, 44, 47, 49, 57, 59, 72, 76, 92, 94, 96, 97, 98, 99, 100, 101, 106, 117, 118, 126, 127, 137, 139, 142, 144, 149, 177, 179, 180, 181, 182, 196, 200, 201, 208, 210, 211, 214, 218, 227, 232, 234, 235, 236, 243, 245, 247, 249, 250, 253, 263, 268, 269, 273, 277, 283, 287, 305, 306, 310, 311, 313, 314, 352, 367, 378, 379, 380, 381, 382, 397, 401, 402, 432, 433, 434, 435, 436, 437, 451, 454, 456, 459, 464, 469, 474, 484, 487, 488, 489, 490, 491, 506, 507, 510, 511, 512, 514, 515
MDA _{Bal.}	389	1, 4, 7, 8, 12, 15, 16, 17, 18, 19, 20, 22, 25, 26, 27, 28, 29, 30, 35, 36, 37, 38, 39, 40, 44, 45, 46, 47, 48, 49, 57, 58, 59, 60, 62, 65, 66, 67, 70, 72, 73, 74, 75, 76, 77, 78, 79, 88, 89, 90, 91, 92, 93, 94, 96, 97, 98, 99, 100, 101, 102, 103, 105, 106, 107, 109, 110, 111, 112, 113, 114, 116, 117, 118, 119, 121, 122, 123, 124, 125, 126, 127, 128, 135, 137, 138, 139, 140, 141, 142, 143, 144, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 158, 159, 162, 163, 164, 166, 168, 170, 172, 174, 176, 177, 178, 179, 180, 181, 182, 184, 186, 187, 188, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 205, 206, 208, 209, 210, 211, 212, 214, 216, 218, 219, 220, 221, 222, 224, 226, 227, 228, 230, 231, 232, 233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 245, 247, 249, 250, 251, 252, 253, 254, 255, 256, 257, 258, 259, 260, 261, 263, 264, 265, 267, 268, 269, 271, 273, 275, 276, 277, 279, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 296, 297, 298, 300, 301, 302, 303, 304, 305, 306, 308, 310, 311, 312, 313, 314, 315, 316, 317, 318, 319, 321, 322, 323, 324, 327, 329, 330, 333, 334, 335, 336, 337, 338, 340, 341, 345, 347, 348, 349, 350, 351, 352, 353, 354, 355, 356, 357, 358, 359, 360, 361, 362, 363, 366, 367, 368, 370, 372, 373, 374, 375, 376, 377, 378, 379, 380, 381, 382, 383, 384, 385, 387, 389, 396, 397, 399, 400, 401, 402, 403, 404, 405, 406, 407, 409, 412, 413, 415, 417, 419, 420, 421, 422, 425, 427, 428, 429, 431, 432, 433, 434, 435, 436, 437, 438, 439, 440, 441, 442, 443, 451, 452, 453, 454, 455, 456, 457, 459, 460, 461, 462, 464, 465, 469, 470, 472, 474, 475, 476, 478, 480, 482, 483, 484, 485, 486, 487, 488, 489, 490, 491, 492, 493, 494, 495, 496, 497, 498, 504, 506, 507, 509, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520, 521, 522, 523, 524, 525, 528, 530, 534, 535, 536, 537, 538, 539, 542, 546, 548, 549
MDG _{Bal.}	169	1, 4, 12, 16, 17, 19, 20, 22, 26, 30, 44, 48, 49, 58, 59, 73, 74, 75, 76, 78, 79, 89, 91, 92, 94, 96, 97, 98, 99, 101, 105, 106, 110, 112, 114, 116, 117, 118, 119, 121, 122, 124, 125, 126, 137, 139, 140, 141, 142, 144, 150, 151, 152, 153, 156, 157, 163, 177, 179, 181, 182, 195, 196, 201, 203, 205, 208, 209, 210, 214, 216, 218, 226, 227, 231, 233, 234, 235, 236, 237, 239, 243, 245, 247, 249, 253, 254, 258, 261, 263, 268, 269, 273, 275, 277, 279, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 297, 298, 300, 304, 305, 306, 310, 311, 315, 316, 318, 321, 322, 323, 324, 329, 334, 336, 337, 341, 345, 347, 348, 354, 356, 357, 360, 361, 367, 380, 381, 382, 383, 397, 401, 402, 412, 428, 433, 436, 437, 451, 455, 457, 459, 469, 476, 478, 487, 488, 489, 490, 491, 492, 493, 494, 506, 507, 511, 512, 519, 520, 530

B.1.5. Day 6 (16/Jun)

Features Set	Number of Features	Features Indexes
MDA	60	7, 12, 15, 19, 26, 27, 28, 29, 30, 36, 38, 40, 47, 59, 62, 63, 64, 67, 73, 75, 77, 79, 89, 90, 91, 96, 97, 100, 105, 109, 110, 111, 116, 119, 126, 127, 142, 150, 152, 153, 156, 157, 226, 241, 269, 273, 275, 281, 287, 288, 290, 329, 345, 352, 360, 361, 520, 530, 535, 548
MDG	70	4, 7, 12, 15, 16, 17, 18, 20, 22, 25, 26, 28, 29, 36, 37, 40, 44, 45, 47, 57, 59, 62, 63, 64, 67, 72, 74, 75, 79, 89, 90, 91, 92, 94, 96, 97, 98, 100, 101, 105, 106, 109, 110, 111, 114, 117, 118, 119, 121, 126, 127, 137, 140, 142, 143, 157, 264, 268, 269, 273, 275, 277, 286, 287, 292, 293, 294, 300, 329, 530
MDA _{Bal.}	171	1, 4, 7, 8, 15, 16, 19, 22, 25, 29, 30, 35, 36, 37, 38, 40, 44, 45, 47, 48, 49, 55, 57, 59, 62, 63, 64, 72, 73, 75, 76, 77, 79, 88, 89, 90, 91, 92, 96, 97, 98, 99, 100, 101, 103, 104, 105, 108, 109, 110, 111, 112, 114, 116, 117, 118, 119, 121, 122, 124, 126, 127, 137, 138, 139, 140, 141, 142, 143, 144, 146, 147, 148, 149, 150, 151, 152, 153, 154, 155, 156, 157, 205, 209, 210, 216, 222, 226, 227, 231, 232, 233, 234, 235, 236, 237, 239, 241, 243, 245, 247, 249, 251, 253, 256, 258, 259, 261, 263, 264, 265, 268, 269, 273, 275, 277, 279, 281, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 297, 298, 299, 300, 302, 303, 304, 305, 306, 308, 310, 311, 312, 314, 318, 319, 321, 329, 338, 341, 345, 350, 354, 355, 357, 359, 360, 361, 465, 469, 480, 487, 488, 489, 498, 507, 519, 520, 523, 530, 535, 542, 546
MDG _{Bal.}	84	4, 7, 12, 30, 36, 45, 62, 63, 64, 76, 77, 88, 89, 90, 91, 96, 97, 99, 104, 109, 110, 111, 114, 126, 127, 137, 138, 141, 143, 144, 147, 150, 151, 152, 153, 157, 209, 226, 231, 232, 233, 234, 235, 241, 247, 249, 253, 258, 261, 263, 265, 269, 275, 277, 279, 284, 286, 287, 288, 290, 291, 292, 293, 295, 297, 298, 310, 312, 313, 315, 321, 324, 341, 345, 349, 354, 360, 361, 469, 480, 506, 520, 542, 546

B.1.6. Day 7 (17/Jun)

Features Set	Number of Features	Features Indexes
MDA	355	4, 7, 12, 17, 20, 21, 25, 27, 29, 30, 35, 36, 38, 41, 42, 44, 45, 46, 47, 51, 52, 53, 54, 55, 56, 59, 61, 62, 63, 64, 65, 66, 72, 73, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 89, 90, 91, 94, 95, 96, 97, 98, 99, 100, 105, 107, 110, 115, 116, 119, 120, 121, 123, 129, 130, 132, 133, 134, 138, 139, 141, 143, 144, 145, 146, 147, 148, 149, 153, 154, 156, 158, 160, 161, 162, 163, 165, 166, 167, 168, 169, 171, 173, 175, 177, 178, 180, 181, 182, 185, 186, 189, 190, 191, 192, 193, 194, 195, 196, 197, 198, 199, 200, 201, 202, 203, 204, 207, 209, 210, 211, 212, 213, 215, 217, 218, 221, 222, 223, 225, 226, 228, 229, 230, 231, 232, 233, 236, 237, 241, 244, 246, 248, 252, 253, 255, 256, 257, 258, 260, 261, 262, 263, 264, 265, 266, 267, 268, 269, 270, 273, 275, 276, 277, 278, 279, 280, 281, 282, 283, 284, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 297, 298, 299, 300, 301, 302, 303, 304, 305, 306, 308, 309, 310, 311, 312, 317, 318, 319, 320, 322, 323, 324, 325, 326, 327, 328, 329, 331, 332, 333, 338, 339, 340, 341, 342, 343, 344, 345, 346, 348, 351, 352, 355, 358, 360, 362, 363, 364, 365, 366, 367, 369, 370, 371, 374, 377, 380, 381, 382, 383, 384, 390, 391, 392, 393, 394, 395, 397, 399, 400, 402, 403, 404, 405, 407, 408, 409, 410, 411, 412, 414, 416, 417, 418, 419, 421, 422, 423, 424, 426, 427, 429, 430, 431, 433, 434, 436, 437, 440, 442, 444, 445, 446, 447, 448, 449, 450, 452, 453, 456, 457, 462, 463, 464, 466, 467, 469, 471, 472, 473, 474, 475, 477, 478, 479, 481, 484, 485, 486, 489, 490, 491, 492, 496, 498, 499, 500, 501, 502, 503, 504, 505, 506, 507, 509, 511, 513, 515, 517, 518, 521, 526, 527, 528, 529, 530, 531, 532, 533, 536, 537, 538, 539, 540, 541, 542, 543, 544, 545, 547, 549
MDG	138	1, 4, 7, 12, 20, 36, 44, 45, 59, 62, 63, 64, 75, 77, 78, 79, 89, 90, 94, 97, 98, 101, 105, 106, 109, 110, 111, 122, 123, 140, 141, 144, 148, 152, 158, 162, 163, 166, 168, 178, 179, 180, 181, 182, 195, 196, 198, 199, 200, 201, 202, 203, 204, 212, 218, 222, 230, 231, 232, 233, 237, 245, 251, 252, 254, 255, 256, 259, 263, 264, 268, 269, 273, 275, 277, 281, 283, 284, 286, 287, 288, 289, 292, 293, 294, 295, 297, 298, 300, 302, 304, 306, 308, 309, 311, 312, 313, 314, 323, 329, 335, 336, 345, 348, 355, 362, 366, 367, 370, 379, 381, 382, 383, 397, 399, 400, 402, 403, 404, 417, 431, 437, 453, 469, 476, 485, 489, 491, 493, 496, 497, 507, 509, 513, 536, 538, 546, 548
MDA _{Bal.}	323	1, 4, 7, 12, 16, 20, 21, 26, 28, 29, 30, 35, 36, 37, 41, 42, 44, 47, 49, 51, 52, 53, 54, 55, 56, 59, 61, 62, 63, 64, 66, 70, 72, 76, 77, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 90, 94, 95, 97, 98, 99, 105, 107, 110, 112, 113, 114, 115, 116, 117, 119, 120, 121, 129, 131, 132, 133, 142, 143, 144, 145, 149, 150, 151, 152, 153, 154, 155, 157, 158, 160, 161, 163, 165, 167, 169, 171, 173, 175, 177, 178, 179, 180, 181, 182, 188, 190, 191, 193, 194, 195, 196, 197, 198, 200, 202, 203, 206, 208, 209, 210, 211, 213, 214, 215, 217, 218, 219, 221, 222, 223, 224, 225, 226, 229, 230, 231, 232, 233, 234, 235, 236, 237, 239, 241, 243, 244, 245, 246, 247, 248, 249, 251, 260, 261, 263, 264, 265, 266, 268, 269, 270, 273, 275, 277, 278, 279, 280, 281, 282, 283, 285, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 297, 298, 300, 302, 303, 304, 305, 306, 311, 312, 313, 314, 316, 318, 319, 322, 323, 324, 325, 326, 327, 328, 329, 331, 332, 336, 338, 339, 340, 342, 343, 344, 346, 355, 356, 357, 361, 363, 364, 365, 367, 368, 369, 371, 373, 374, 377, 378, 379, 380, 381, 382, 383, 385, 390, 391, 392, 393, 394, 395, 396, 401, 402, 403, 404, 408, 410, 411, 412, 414, 416, 418, 420, 422, 423, 424, 425, 426, 429, 430, 431, 435, 436, 437, 438, 444, 445, 446, 447, 448, 449, 450, 451, 453, 461, 462, 463, 464, 466, 467, 471, 472, 473, 475, 476, 477, 478, 479, 480, 481, 483, 484, 485, 486, 488, 489, 491, 495, 497, 498, 501, 502, 503, 504, 505, 506, 507, 509, 511, 513, 514, 516, 518, 522, 525, 526, 527, 528, 529, 530, 531, 532, 533, 537, 540, 541, 542, 543, 544, 545, 547
MDG _{Bal.}	135	1, 4, 7, 15, 20, 26, 35, 48, 49, 57, 59, 61, 62, 63, 64, 69, 70, 71, 74, 76, 77, 79, 90, 91, 99, 109, 110, 111, 113, 114, 116, 119, 121, 128, 137, 140, 142, 146, 151, 152, 158, 178, 180, 181, 182, 196, 201, 204, 205, 209, 210, 222, 226, 235, 237, 238, 241, 242, 250, 255, 261, 263, 264, 265, 269, 275, 277, 279, 281, 286, 287, 288, 289, 290, 291, 292, 293, 294, 295, 297, 300, 302, 304, 305, 306, 310, 311, 312, 315, 316, 321, 323, 329, 336, 337, 341, 347, 350, 356, 357, 361, 367, 370, 379, 380, 381, 383, 402, 403, 404, 428, 431, 432, 434, 437, 462, 464, 474, 476, 480, 488, 489, 490, 491, 495, 497, 507, 512, 513, 514, 515, 519, 523, 537, 548

B.2. Models Results

This section lists the results of the model evaluations of the experiments discussed in **Section 6.5** (see **Chapter 6**).

B.2.1. C5.0 (Decision Trees) Results

Table B.1 presents the results of the experiments using the **C5.0 algorithm**. Each *shaded* cell of **Table B.1** contains the maximum G-Mean Accuracy achieved at the CV stage, where the model's threshold was set. Every other cell contains two performance measures. The top measure is the model's performance on the test subset (the day file) when its optimal (CV) cutoff was used and the second measure is the model's performance when the cutoff was adapted for the test data. The measure in bold is the greater of the two measures.

Figure B.1 shows the performance (G-Mean Accuracy) for each model for each training day for the **C5.0 algorithm**. Each plot shows models performances (in every sub-plot) under different feature sets (Full, MDA, MDG, MDA_{Bal.} and MDG_{Bal.}) and data balances (Original and Balanced). Every sub-plot illustrates the G-Mean Accuracy for that day's model after being evaluated using all the other days' files (along the x-axis). For each evaluation there are two G-Mean Accuracy readings; one is based on the model's optimal threshold ('CV Cutoff' in red colour) while the other uses the adapted threshold ('Adp. Cutoff' in blue colour) on the test data. The first day (along the x-axis) matches the training day of the main plot and corresponds to the CV results of that model.

		Original						Balance					
		Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7
Full	MDL 2	0.9996	0.0099	0.0176	0.0053	0.8062	0.0568	0.9999	0.0100	0.0241	0.0053	0.8995	0.0562
			0.5882	0.5045	0.0148	0.9999	0.9934	0.9044	0.6387	0.0144	1.0000	0.9879	
	MDL 3	0.5967	0.9834	0.8106	0.0105	0.0000	0.9477	0.9014	0.9823	0.8661	0.0130	0.6304	0.7753
		0.9745		0.9137	0.0211	0.5650	0.9776	0.9487		0.9043	0.0134	0.9652	0.8440
	MDL 4	0.0375	0.9373	0.9813	0.0000	0.0000	0.0316	0.9916	0.9249	0.9815	0.0129	0.9985	0.7109
		0.9937	0.9507		0.0130	0.9514	0.0616	0.9931	0.9271		0.0176	0.9989	0.9900
	MDL 5	0.9982	0.0000	0.0216	0.9977	0.9045	0.0875	0.8820	0.0100	0.3610	0.9975	0.8523	0.7759
	0.9993	0.1950	0.4956		0.9999	0.9895	0.9934	0.1419	0.6242		0.9966	0.9907	
MDL 6	0.0000	0.0000	0.0000	0.0000	0.4606	0.0000	0.9910	0.0200	0.0250	0.0479	1.0000	0.7458	
	0.0000	0.0000	0.0000	0.0000		0.0000	0.9988	0.1485	0.0278	0.0480		0.9980	
MDL 7	0.9901	0.9151	0.4357	0.0043	0.9998	0.9999	0.9956	0.3826	0.3916	0.0092	0.9999	1.0000	
	0.9945	0.9413	0.7629	0.0132	0.9998	0.9999	0.9962	0.9210	0.6703	0.0258	1.0000		
MDA	MDL 2	0.9998	0.0000	0.0279	0.0479	0.8528	0.0568	1.0000	0.0000	0.0279	0.0479	0.8528	0.0568
			0.0940	0.0279	0.0479	0.8528	0.0568		0.0940	0.0279	0.0479	0.8528	0.0568
	MDL 3	0.9814	0.9838	0.5062	0.2146	0.7327	0.1403	0.5278	0.9826	0.8504	0.8807	0.8344	0.8673
		0.9845		0.5720	0.7332	0.9890	0.4531	0.7520		0.8558	0.9420	0.8395	0.9405
	MDL 4	0.1127	0.8999	0.9802	0.0402	0.3013	0.0283	0.7905	0.9193	0.9800	0.0790	0.9985	0.0647
		0.6653	0.9476		0.3447	0.8502	0.1108	0.9882	0.9299		0.3659	0.9992	0.4175
	MDL 5	0.9902	0.0100	0.0983	0.9977	0.8528	0.0550	0.9087	0.0100	0.6697	0.9974	0.9478	0.8538
	0.9990	0.4568	0.4944		0.9504	0.1438	0.9857	0.0378	0.8277		0.9896	0.9923	
MDL 6	0.0000	0.0000	0.0000	0.0000	0.4606	0.0000	0.9808	0.0141	0.0128	0.0473	1.0000	0.0532	
	0.0000	0.0000	0.0000	0.0000		0.0000	1.0000	0.8035	0.4143	0.7486		0.9999	
MDL 7	0.9923	0.8980	0.4348	0.0485	0.9998	0.9998	0.9469	0.1264	0.0249	0.7837	1.0000	0.9998	
	0.9976	0.9286	0.4498	0.9837	1.0000	0.9998	0.9940	0.9236	0.8137	0.9907	1.0000		
MDG	MDL 2	0.9998	0.0000	0.0279	0.0479	0.8528	0.0568	1.0000	0.0000	0.0279	0.0479	0.8528	0.0568
			0.0940	0.0279	0.0479	0.8528	0.0568		0.0940	0.0279	0.0479	0.8528	0.0568
	MDL 3	0.1309	0.9702	0.8138	0.5949	0.0000	0.6922	0.4031	0.9656	0.8976	0.5522	0.5120	0.7539
		0.8279		0.8979	0.7917	0.4171	0.7796	0.8978		0.9092	0.6895	0.7127	0.8859
	MDL 4	0.6496	0.9160	0.9431	0.9906	0.0000	0.9976	0.5688	0.8835	0.9165	0.9049	0.0000	0.9666
		0.9799	0.9246		0.9912	0.9815	0.9977	0.8639	0.9026		0.9100	0.0000	0.9792
	MDL 5	0.9208	0.0200	0.0176	0.9969	0.0000	0.9611	0.9927	0.0141	0.3249	0.9972	0.8519	0.9978
	0.9896	0.2523	0.8186		0.9529	0.9836	0.9961	0.8160	0.8821		0.9965	0.9982	
MDL 6	0.0000	0.0000	0.0000	0.0000	0.4606	0.0000	0.0614	0.0100	0.0000	0.0092	1.0000	0.0142	
	0.0000	0.0000	0.0000	0.0000		0.0000	0.9935	0.8094	0.4154	0.0822		0.9993	
MDL 7	0.9943	0.8910	0.4342	0.0485	0.9999	0.9999	0.9867	0.9068	0.4390	0.5440	0.9525	0.9997	
	0.9976	0.9253	0.4423	0.9837	1.0000	0.9999	0.9944	0.9142	0.6559	0.9853	0.9962		
MDA _{Bal.}	MDL 2	0.9998	0.0000	0.0279	0.0479	0.8528	0.0568	1.0000	0.0000	0.0279	0.0479	0.8528	0.0568
			0.0940	0.0279	0.0479	0.8528	0.0568		0.0940	0.0279	0.0479	0.8528	0.0568
	MDL 3	0.9849	0.9833	0.5069	0.8782	0.5156	0.4736	0.7077	0.9821	0.6074	0.5015	0.7124	0.0816
		0.9876		0.5967	0.9296	0.9843	0.8484	0.8853		0.8378	0.8607	0.7731	0.4364
	MDL 4	0.1337	0.9435	0.9802	0.0580	0.3013	0.0245	0.9852	0.6290	0.9805	0.0627	0.9958	0.4256
		0.6646	0.9492		0.9824	0.8496	0.1062	0.9906	0.9232		0.1455	0.9971	0.6606
	MDL 5	0.9984	0.1142	0.0993	0.9977	0.7977	0.0568	0.8156	0.0100	0.0278	0.9974	0.7368	0.3558
	0.9995	0.4586	0.7236		0.9367	0.6729	0.9888	0.1569	0.6789		0.9954	0.9915	
MDL 6	0.0000	0.0000	0.0000	0.0000	0.4606	0.0000	0.0614	0.0100	0.0125	0.0092	1.0000	0.0142	
	0.0000	0.0000	0.0000	0.0000		0.0000	0.9985	0.8062	0.4115	0.7465		0.9996	
MDL 7	0.9923	0.8952	0.4348	0.0485	0.9998	0.9998	0.9873	0.1978	0.0892	0.7895	0.9999	0.9998	
	0.9976	0.9285	0.4445	0.9829	1.0000	0.9998	0.9940	0.9236	0.8137	0.9907	1.0000		
MDG _{Bal.}	MDL 2	0.9998	0.0000	0.0279	0.0479	0.8528	0.0568	1.0000	0.0000	0.0279	0.0479	0.8528	0.0568
			0.0940	0.0279	0.0479	0.8528	0.0568		0.0940	0.0279	0.0479	0.8528	0.0568
	MDL 3	0.2910	0.9827	0.6145	0.1415	0.0000	0.4229	0.8819	0.9820	0.8189	0.9222	0.5312	0.6688
		0.4477		0.8308	0.7589	0.0000	0.8031	0.8896		0.8388	0.9300	0.5783	0.8254
	MDL 4	0.6499	0.9485	0.9764	0.3011	0.0567	0.9824	0.9265	0.9265	0.9803	0.7625	0.8510	0.0615
		0.9342	0.9497		0.9799	0.9918	0.3565	0.9826			0.9326	0.9341	0.0756
	MDL 5	0.9673	0.0158	0.1048	0.9976	0.7977	0.0531	0.8201	0.0000	0.0736	0.9973	0.8510	0.4139
	0.9985	0.4339	0.4977		0.9522	0.0680	0.9485	0.1562	0.7646		0.9005	0.8068	
MDL 6	0.0000	0.0000	0.0000	0.0000	0.4606	0.0000	0.0614	0.0100	0.0000	0.0092	1.0000	0.0142	
	0.0000	0.0000	0.0000	0.0000		0.0000	0.9935	0.8094	0.4154	0.0822		0.9993	
MDL 7	0.9929	0.1801	0.0729	0.7814	0.9997	0.9997	0.9883	0.0064	0.0139	0.0533	0.9998	0.9999	
	0.9954	0.9267	0.4813	0.9778	1.0000	0.9998	0.9900	0.9054	0.5412	0.9913	1.0000		

Table B.1: The performance of models (G-Mean Accuracy) for the original and adapted cutoff (threshold) for the C5.0 algorithm.

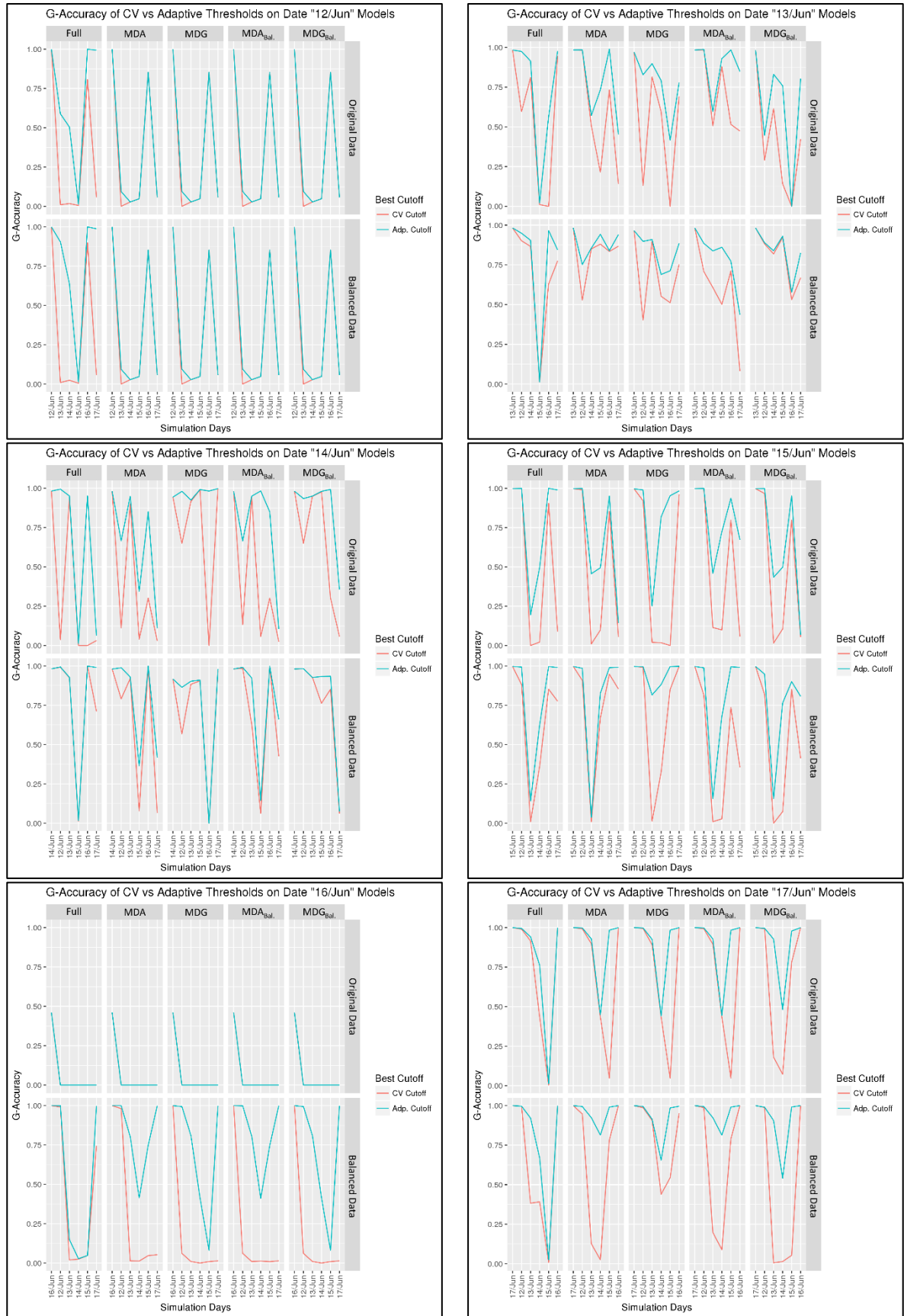


Figure B.1: Comparison plot of the performance of C5.0 models (G-Mean Accuracy) for every training day in the STA2018 dataset between the optimal (CV) and adaptive cutoffs.

B.2.2. Random Forest (RF) Results

Table B.2 presents the results of the experiments using the **Random Forest (RF) algorithm**. Each *shaded* cell of **Table B.2** contains the maximum G-Mean Accuracy achieved at the CV stage when the model's threshold was set. Every other cell contains two performance measures. The top measure is the model's performance on the test subset (the day file) when its optimal (CV) cutoff was used and the second one is the performance when the cutoff was adapted to the test data. The measure in bold is the greater of the two measures.

Figure B.2 shows the performance (G-Mean Accuracy) for each model for each training day for the **Random Forest (RF) algorithm**. Each plot shows models performances (in every sub-plot) under different feature sets (Full, MDA, MDG, MDA_{Bal.} and MDG_{Bal.}) and data balances (Original and Balanced). Every sub-plot illustrates the G-Mean Accuracy for that day's model after being evaluated using all the other days' files (along the x-axis). For each evaluation there are two G-Mean Accuracy readings; one is based on the model's optimal threshold ('CV Cutoff' in red colour) while the other uses the adapted threshold ('Adp. Cutoff' in blue colour) on the test data. The first day (along the x-axis) matches the training day of the main plot and corresponds to the CV results of that model.

		Original						Balance					
		Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7
		Full	MDL 2	1.0000	0.0042 0.9272	0.0279 0.9478	0.0482 0.9094	1.0000 1.0000	0.0602 0.9987	1.0000	0.0100 0.9293	0.0216 0.9413	0.0476 0.9257
MDL 3	0.9521 0.9932		0.9849	0.9213 0.9739	0.6884 0.9577	0.8560 0.9870	0.9411 0.9976	0.9663 0.9925	0.9848	0.9191 0.9688	0.7498 0.9340	0.9530 0.9793	0.9758 0.9987
MDL 4	0.9428 0.9945		0.9301 0.9560	0.9827	0.8425 0.9920	0.8832 0.9998	0.9404 0.9978	0.9762 0.9948	0.9237 0.9471	0.9829	0.8991 0.9858	0.9268 0.9997	0.9688 0.9882
MDL 5	0.9997 0.9999		0.0133 0.9205	0.0648 0.9395	0.9978	0.9535 1.0000	0.0585 0.9976	0.9888 0.9971	0.2259 0.9156	0.6808 0.9302	0.9981	0.9934 0.9998	0.9925 0.9926
MDL 6	0.9912 1.0000		0.0000 0.9120	0.0250 0.5737	0.0479 0.9671	1.0000	0.0568 0.9998	0.0217 0.9999	0.0100 0.8685	0.0125 0.6837	0.0053 0.1048	1.0000	0.0142 0.9998
MDL 7	0.9964 0.9979		0.9140 0.9351	0.4314 0.9340	0.7864 0.9911	1.0000 1.0000	1.0000	0.0217 0.9998	0.0100 0.9381	0.0125 0.9319	0.0341 0.9755	0.5279 1.0000	1.0000
MDA	MDL 2		1.0000	0.0463 0.9308	0.0279 0.9417	0.0482 0.9322	1.0000 1.0000	0.0585 0.9983	1.0000	0.0100 0.9281	0.0216 0.9504	0.0476 0.7999	0.8528 1.0000
	MDL 3	0.9790 0.9895	0.9848	0.9636 0.9738	0.9089 0.9490	0.9570 0.9844	0.9890 0.9963	0.9683 0.9927	0.9848	0.9197 0.9690	0.8158 0.9336	0.8781 0.9839	0.9282 0.9983
	MDL 4	0.9493 0.9947	0.9374 0.9557	0.9826	0.8913 0.9917	0.8956 0.9971	0.9448 0.9984	0.9463 0.9948	0.9269 0.9449	0.9827	0.8882 0.9889	0.9024 0.9993	0.9467 0.9942
	MDL 5	0.9997 0.9999	0.0141 0.9162	0.0872 0.9388	0.9978	0.9535 1.0000	0.0619 0.9951	0.9891 0.9996	0.2439 0.9218	0.6620 0.9411	0.9981	0.9935 0.9981	0.9933 0.9935
	MDL 6	0.9983 1.0000	0.0100 0.8553	0.0250 0.4649	0.0479 0.3326	1.0000	0.0568 0.9999	0.0217 1.0000	0.0100 0.9320	0.0176 0.4956	0.0053 0.1731	1.0000	0.0142 1.0000
	MDL 7	0.9933 0.9971	0.9309 0.9360	0.4459 0.9370	0.9875 0.9911	1.0000 1.0000	1.0000	0.0217 0.9998	0.0100 0.9355	0.0125 0.9280	0.0367 0.9699	0.4053 1.0000	1.0000
	MDG	MDL 2	1.0000	0.0452 0.9296	0.0279 0.9317	0.0482 0.8425	1.0000 1.0000	0.0585 0.9982	1.0000	0.0100 0.9287	0.0216 0.9457	0.0473 0.9430	0.8528 1.0000
MDL 3		0.5035 0.8340	0.9780	0.9171 0.9185	0.6307 0.8195	0.0000 0.7887	0.7868 0.9471	0.5308 0.8000	0.9698	0.9051 0.9112	0.5622 0.7204	0.0000 0.7087	0.6543 0.8924
MDL 4		0.7032 0.7703	0.9160 0.9242	0.9432	0.9856 0.9913	0.0000 0.8236	0.9884 0.9974	0.6800 0.8676	0.8996 0.9000	0.8906	0.9829 0.7499	0.0000 0.9538	0.9531
MDL 5		0.1390 0.9992	0.0100 0.8518	0.0254 0.9138	0.9978	0.6742 0.9996	0.0375 0.9976	0.9897 0.9926	0.0223 0.8016	0.5450 0.8817	0.9981	0.9999 1.0000	0.3540 0.9982
MDL 6		0.9982 1.0000	0.0000 0.8573	0.0250 0.4349	0.0479 0.6449	1.0000	0.0568 0.9998	0.0217 1.0000	0.0100 0.8710	0.0125 0.7446	0.0053 0.0767	1.0000	0.0142 1.0000
MDL 7		0.9926 0.9972	0.9353 0.9357	0.4493 0.9354	0.9911 0.9912	0.9999 1.0000	1.0000	0.0217 1.0000	0.0100 0.9376	0.0125 0.9377	0.0136 0.9700	0.4116 1.0000	1.0000
MDABal.		MDL 2	1.0000	0.0418 0.9331	0.0279 0.9434	0.0482 0.9739	1.0000 1.0000	0.0586 0.9985	1.0000	0.0100 0.9277	0.0216 0.9460	0.0473 0.9603	0.8528 1.0000
	MDL 3	0.9758 0.9933	0.9848	0.9558 0.9747	0.8755 0.9604	0.8884 0.9826	0.9777 0.9965	0.9742 0.9919	0.9849	0.9122 0.9715	0.8515 0.9103	0.8980 0.9833	0.9522 0.9983
	MDL 4	0.9505 0.9947	0.9349 0.9560	0.9824	0.8801 0.9913	0.8903 0.9960	0.9464 0.9982	0.9508 0.9945	0.9275 0.9442	0.9827	0.8870 0.9897	0.8912 0.9991	0.9422 0.9864
	MDL 5	0.9996 0.9999	0.0141 0.9163	0.0671 0.9402	0.9978	0.9535 0.9999	0.0619 0.9984	0.9868 0.9973	0.2331 0.9141	0.6778 0.9344	0.9981	0.9933 0.9969	0.9934 0.9941
	MDL 6	0.9947 1.0000	0.0000 0.8610	0.0250 0.4367	0.0479 0.9482	1.0000	0.0568 0.9998	0.0217 1.0000	0.0100 0.8753	0.0125 0.4526	0.0053 0.3582	1.0000	0.0142 1.0000
	MDL 7	0.9955 0.9976	0.9148 0.9366	0.4329 0.9349	0.9083 0.9907	1.0000 1.0000	1.0000	0.0217 0.9997	0.0100 0.9335	0.0125 0.9364	0.0465 0.9680	0.9535 1.0000	1.0000
	MDGBal.	MDL 2	1.0000	0.0457 0.9257	0.0279 0.9409	0.0482 0.9439	1.0000 1.0000	0.0585 0.9983	1.0000	0.0100 0.9272	0.0216 0.9512	0.0473 0.8091	0.8528 1.0000
MDL 3		0.9795 0.9849	0.9850	0.9346 0.9556	0.8227 0.8257	0.8250 0.9309	0.8990 0.9231	0.9710 0.9892	0.9850	0.8950 0.9517	0.7824 0.8583	0.7985 0.9215	0.8641 0.9974
MDL 4		0.9571 0.9923	0.9339 0.9540	0.9827	0.8879 0.9696	0.8889 0.9346	0.9422 0.9962	0.9438 0.9947	0.9313 0.9411	0.9826	0.8991 0.9752	0.9096 0.9521	0.9547 0.9949
MDL 5		0.9975 0.9993	0.0141 0.8693	0.0330 0.9049	0.9978	0.9535 0.9995	0.0531 0.9954	0.9881 0.9941	0.0903 0.9133	0.6163 0.9340	0.9981	0.9931 0.9951	0.9366 0.9927
MDL 6		0.9952 1.0000	0.0141 0.8642	0.0250 0.4449	0.0479 0.9611	1.0000	0.0568 0.9999	0.0217 1.0000	0.0100 0.8675	0.0125 0.5907	0.0053 0.0747	1.0000	0.0142 1.0000
MDL 7		0.9970 0.9974	0.9036 0.9391	0.4303 0.9365	0.0505 0.9908	1.0000 1.0000	1.0000	0.0217 1.0000	0.0100 0.9339	0.0125 0.9412	0.0053 0.9893	0.4273 1.0000	1.0000

Table B.2: Performance of the models (G-Mean Accuracy) for the original and adapted cutoffs (threshold) for the Random Forest (RF) algorithm.

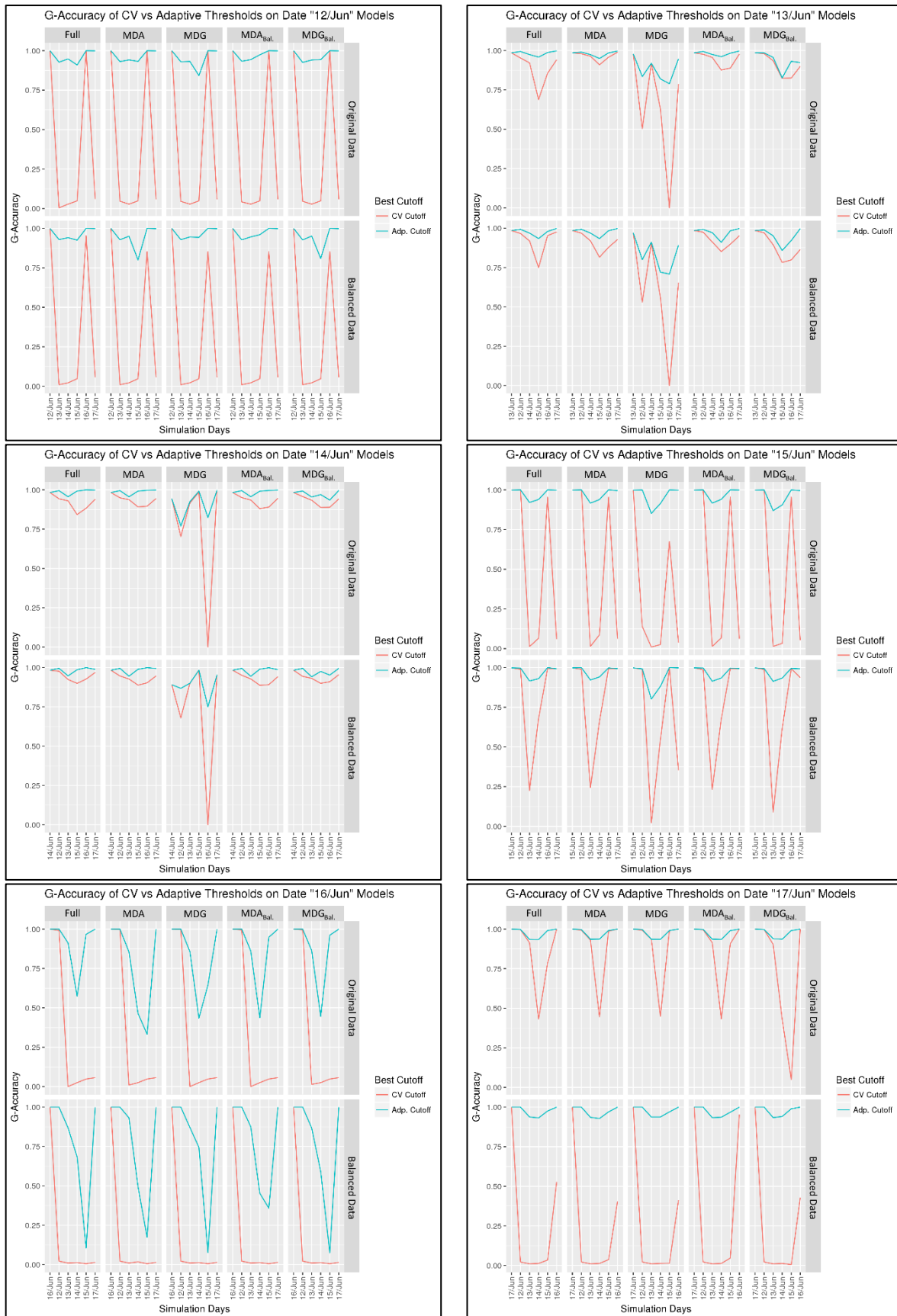


Figure B.2: Comparison plot of the performance of **RF** models (*G*-Mean Accuracy) for every training day in the STA2018 dataset between the optimal (CV) and adaptive cutoffs.

B.2.3. Support Vector Machine (SVM) Results

Table B.3 presents the results of the experiments using the **Support Vector Machine (SVM) algorithm**. Each *shaded* cell of **Table B.3** contains the maximum G-Mean Accuracy reached at the CV stage at which the model's threshold was set. Every other cell contains two performance measures. The top measure is the model's performance on the test subset (day file) when its optimal (CV) cutoff was used and the second one is the performance when the cutoff was adapted to the test data. The measure in bold is the greater of the two measures.

Figure B.3 shows the performance (G-Mean Accuracy) for each model for each training day for the **Support Vector Machine (SVM) algorithm**. Each plot shows models performances (in every sub-plot) under different feature sets (Full, MDA, MDG, MDA_{Bal.} and MDG_{Bal.}) and data balances (Original and Balanced). Every sub-plot illustrates the G-Mean Accuracy for that day's model after being evaluated using all the other days' files (along the x-axis). For each evaluation there are two G-Mean Accuracy readings; one is based on the model's optimal threshold ('CV Cutoff' in red colour) while the other uses the adapted threshold ('Adp. Cutoff' in blue colour) on the test data. The first day (along the x-axis) matches the training day of the main plot and corresponds to the CV results of that model.

		Original						Balance					
		Day 2	Day 3	Day 4	Day 5	Day 6	Day 7	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7
Full	MDL 2	1.0000	0.1630 0.1756	0.3695 0.3977	0.0573 0.1891	0.9857 0.9875	0.9681 0.9703	0.9996	0.9076 0.9137	0.5294 0.5622	0.6040 0.9011	0.9997 0.9999	0.8786 0.9966
	MDL 3	0.5133 0.9312	0.9809	0.5184 0.5200	0.3624 0.8053	0.7800 0.9517	0.8279 0.8661	0.4939 0.8235	0.9754	0.4641 0.6112	0.4042 0.7456	0.4023 0.8289	0.4473 0.8026
	MDL 4	0.3545 0.9397	0.9161 0.9457	0.9814	0.0538 0.3852	0.9182 0.9589	0.4942 0.7560	0.1242 0.9154	0.7312 0.9405	0.9800	0.0160 0.7563	0.0000 0.9677	0.0200 0.8836
	MDL 5	0.0000 0.8674	0.1026 0.1761	0.7030 0.8065	0.9961	0.0000 0.9531	0.7385 0.8429	0.0000 0.9490	0.1387 0.1474	0.5176 0.5208	0.9913	0.4249 0.9242	0.7042 0.7166
	MDL 6	0.9806 0.9913	0.8750 0.8931	0.6296 0.6411	0.0575 0.5101	0.9994	0.9985 0.9990	0.3346 0.9978	0.0282 0.1468	0.0216 0.2367	0.1027 0.9575	1.0000	0.5148 0.8563
	MDL 7	0.9820 0.9906	0.1070 0.2143	0.4882 0.6021	0.0599 0.7417	0.9532 0.9997	0.9999	0.9860 0.9931	0.0489 0.1611	0.1797 0.5437	0.9667 0.9843	0.9995 0.9995	0.9999
	MDA	MDL 2	1.0000	0.9206 0.9224	0.4812 0.7470	0.0542 0.1977	0.9534 0.9999	0.0585 0.9911	0.9997	0.6143 0.6945	0.4374 0.7889	0.2112 0.8507	0.9534 1.0000
MDL 3		0.0000 0.2443	0.9809	0.8553 0.8658	0.0237 0.0382	0.8381 0.9337	0.1051 0.6997	0.9330 0.9621	0.9765	0.8737 0.8743	0.4306 0.8487	0.7185 0.9143	0.8755 0.9184
MDL 4		0.6177 0.9725	0.9447 0.9454	0.9810	0.4665 0.5871	0.9038 0.9978	0.0722 0.7330	0.1466 0.8707	0.9397 0.9413	0.9792	0.9226 0.9360	0.0000 0.9749	0.0317 0.7675
MDL 5		0.0000 0.9743	0.0307 0.1444	0.5224 0.6106	0.9957	0.0000 0.9529	0.0000 0.7213	0.9178 0.9865	0.1268 0.1812	0.5203 0.5285	0.9914	0.0000 0.9433	0.0000 0.4924
MDL 6		0.9953 0.9993	0.8535 0.9134	0.7474 0.8951	0.9685 0.9806	0.9998	0.0568 0.9908	0.9992	0.8966 0.9242	0.3735 0.4774	0.9686 0.9821	1.0000	0.0568 0.9402
MDL 7		0.2622 0.9931	0.0691 0.1308	0.5673 0.6090	0.8781 0.9117	0.9515 0.9971	0.9999	0.9795 0.9922	0.1287 0.1891	0.2820 0.6485	0.9669 0.9902	0.9526 0.9989	0.9995
MDG		MDL 2	1.0000	0.7706 0.7912	0.6099 0.8736	0.0429 0.1953	0.9998 0.9999	0.0602 0.9962	0.9997	0.6713 0.7400	0.4638 0.4888	0.0480 0.4182	0.9534 1.0000
	MDL 3	0.0000 0.5136	0.9391	0.6141 0.6847	0.0000 0.5639	0.0000 0.3726	0.0000 0.3531	0.0000 0.5481	0.9357	0.6555 0.6685	0.0000 0.5900	0.0000 0.3930	0.0000 0.5643
	MDL 4	0.8259 0.9675	0.7480 0.8062	0.8851	0.9870 0.9879	0.9831 0.9927	0.9910 0.9956	0.9773 0.9833	0.8656 0.8665	0.9092	0.9834 0.9884	0.9888 0.9928	0.9888 0.9958
	MDL 5	0.0000 0.9378	0.0331 0.1450	0.7649 0.7727	0.9950	0.9528 0.9967	0.0000 0.3657	0.0803 0.7791	0.0695 0.1176	0.4730 0.4839	0.9849	0.9479 0.9935	0.0647 0.2488
	MDL 6	0.9994 0.9996	0.0895 0.8718	0.0250 0.5703	0.8181 0.9710	0.9999	0.0585 0.8849	0.9916 0.9992	0.0000 0.8828	0.0210 0.4595	0.7853 0.9673	1.0000	0.0550 0.9852
	MDL 7	0.0000 0.9742	0.9120 0.9201	0.4510 0.6205	0.6801 0.7580	0.9519 0.9974	0.9998	0.8579 0.9903	0.9096 0.9100	0.5099 0.6328	0.9655 0.9849	0.9517 0.9982	0.9995
	MDA _{Bal.}	MDL 2	1.0000	0.8373 0.8590	0.6823 0.8670	0.0727 0.2976	0.9999 0.9999	0.0585 0.9943	0.9994	0.9202 0.9307	0.4744 0.5384	0.0480 0.7213	0.9999 1.0000
MDL 3		0.0000 0.3185	0.9807	0.7562 0.7765	0.0237 0.0382	0.8247 0.9415	0.2069 0.7326	0.9366 0.9593	0.9761	0.8781 0.8804	0.4806 0.8301	0.7460 0.9235	0.8900 0.9226
MDL 4		0.8749 0.9727	0.9298 0.9416	0.9810	0.4613 0.5578	0.9512 0.9957	0.0837 0.6791	0.1365 0.8578	0.9384 0.9399	0.9791	0.9123 0.9160	0.0000 0.9818	0.0245 0.7552
MDL 5		0.0000 0.9745	0.0223 0.1443	0.4724 0.5477	0.9957	0.0000 0.9531	0.0000 0.7137	0.9151 0.9868	0.1754 0.2139	0.5767 0.6023	0.9924	0.0000 0.9410	0.0000 0.6056
MDL 6		0.9789 0.9940	0.0479 0.3377	0.5601 0.7603	0.9336 0.9640	1.0000	0.1291 0.9749	0.5956 0.9984	0.7399 0.9135	0.2373 0.4465	0.1566 0.9562	1.0000	0.0492 0.9925
MDL 7		0.0795 0.9902	0.0582 0.1513	0.6014 0.6418	0.3942 0.5782	0.9982 0.9986	0.9999	0.6721 0.9925	0.1356 0.2104	0.4007 0.6328	0.9694 0.9849	0.9529 0.9992	0.9996
MDG _{Bal.}		MDL 2	1.0000	0.9263 0.9311	0.7197 0.9164	0.0474 0.5652	0.9534 0.9999	0.0585 0.9974	0.9995	0.8279 0.9140	0.4580 0.5588	0.0484 0.8278	0.9998 0.9999
	MDL 3	0.8727 0.9800	0.9780	0.6410 0.7587	0.3583 0.5333	0.7586 0.9002	0.8608 0.8995	0.5333 0.8954	0.9695	0.4830 0.4834	0.4616 0.4654	0.6226 0.6894	0.7089 0.7755
	MDL 4	0.1225 0.9823	0.9035 0.9435	0.9801	0.7684 0.9535	0.0000 0.9851	0.0316 0.4872	0.1238 0.8983	0.9446 0.9449	0.9781	0.8969 0.8983	0.0000 0.8981	0.0245 0.6145
	MDL 5	0.0307 0.9694	0.0141 0.1687	0.6289 0.6689	0.9958	0.7384 0.9521	0.0000 0.6616	0.8877 0.9827	0.0518 0.1426	0.0889 0.3074	0.9900	0.0000 0.9714	0.0000 0.5188
	MDL 6	0.9978 0.9996	0.0100 0.2816	0.7001 0.7927	0.5854 0.9595	1.0000	0.1670 0.9908	0.7335 0.9961	0.8746 0.9199	0.3746 0.4683	0.9650 0.9665	1.0000	0.0531 0.9930
	MDL 7	0.0000 0.9320	0.6830 0.7189	0.8442 0.8991	0.5596 0.7260	0.9527 0.9978	0.9998	0.8509 0.9921	0.1288 0.2576	0.2795 0.6870	0.9692 0.9873	0.9990 0.9994	0.9996

Table B.3: The performance of models (G-Mean Accuracy) for the original and adapted cutoffs (thresholds) for SVM algorithm.

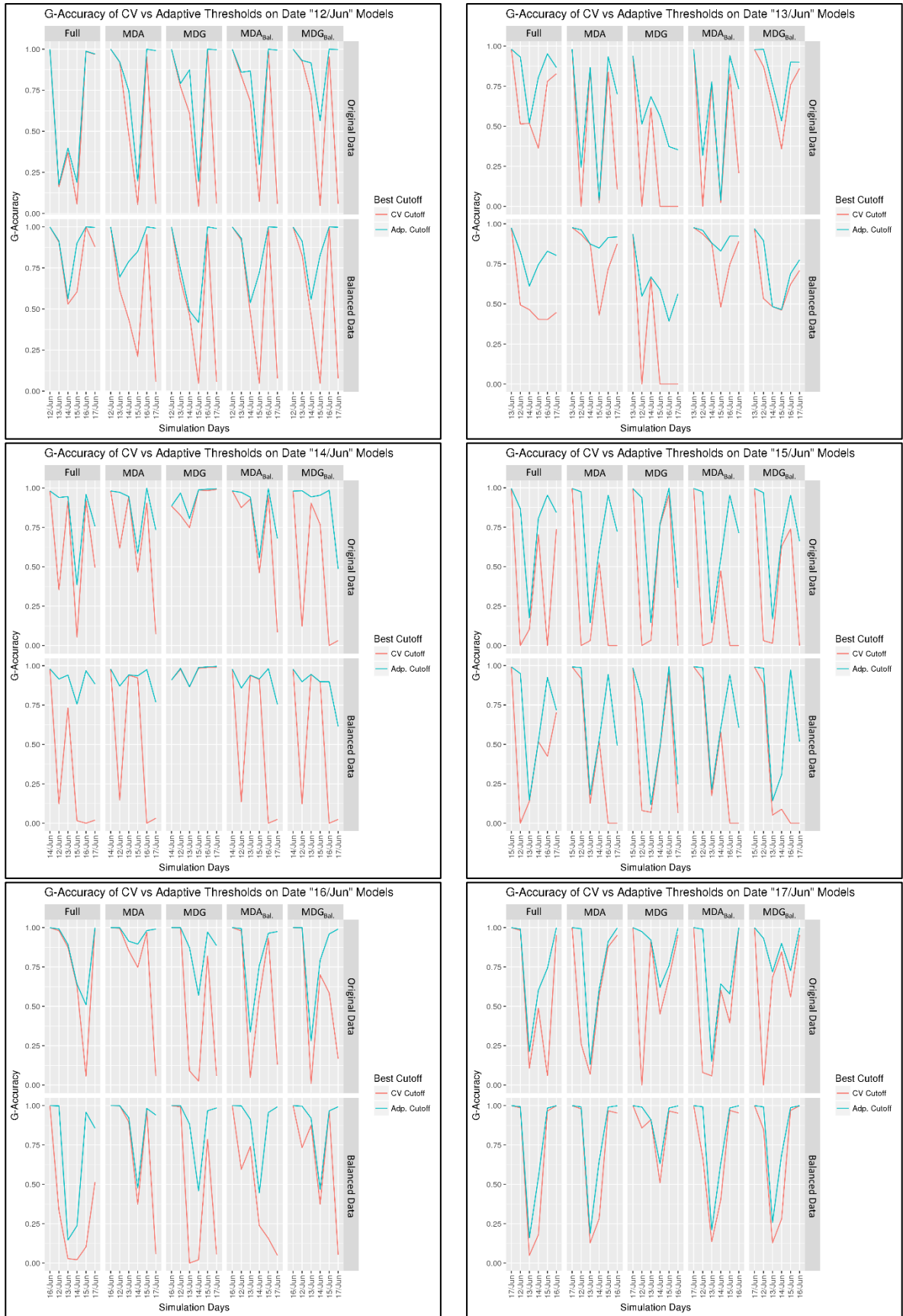


Figure B.3: Comparison plot of the performance of SVM models (G-Mean Accuracy) for every training day in the STA2018 dataset between the optimal (CV) and adaptive cutoffs.



Appendix (C) Results of Chapter 7 (Third) Experiment

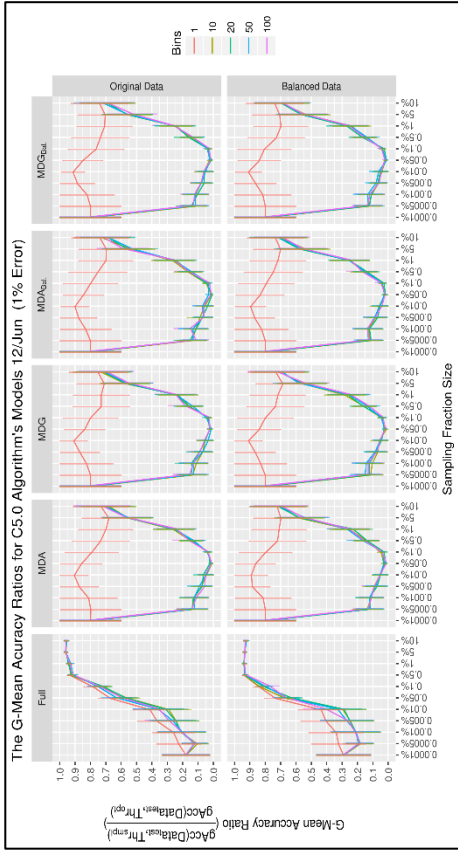
This appendix documents the outcomes of the third set of experiments presented in **Chapter 7**, which discussed the identification of the optimal threshold for an evaluation data using a small subset as a validation dataset.

C.1. Results of Every Day

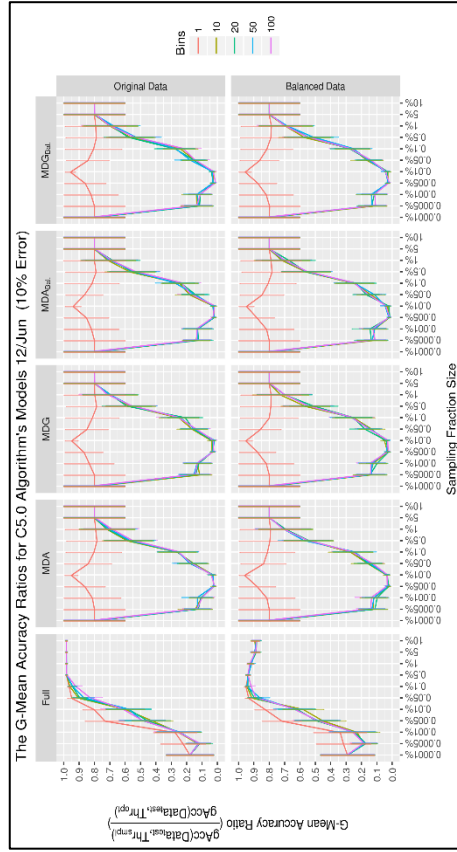
This section lists the result G-Mean Accuracy Ratio (GAR) plots of every simulation day for every ML algorithm. The following **Table C.1** maps the figures to their related ML algorithm and simulation day.

Simulation Day		ML Algorithm		
		C5.0	RF	SVM
Day 2 - Sat	12 Jun	Figure C.1	Figure C.7	Figure C.13
Day 3 - Sun	13 Jun	Figure C.2	Figure C.8	Figure C.14
Day 4 - Mon	14 Jun	Figure C.3	Figure C.9	Figure C.15
Day 5 - Tue	15 Jun	Figure C.4	Figure C.10	Figure C.16
Day 6 - Wed	16 Jun	Figure C.5	Figure C.11	Figure C.17
Day 7 - Thu	17 Jun	Figure C.6	Figure C.12	Figure C.18

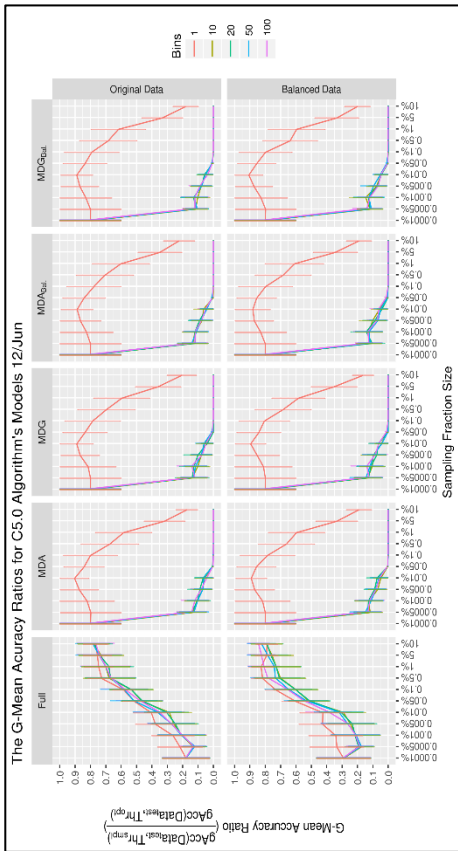
Table C.1: Figures map of the results of the third experiment.



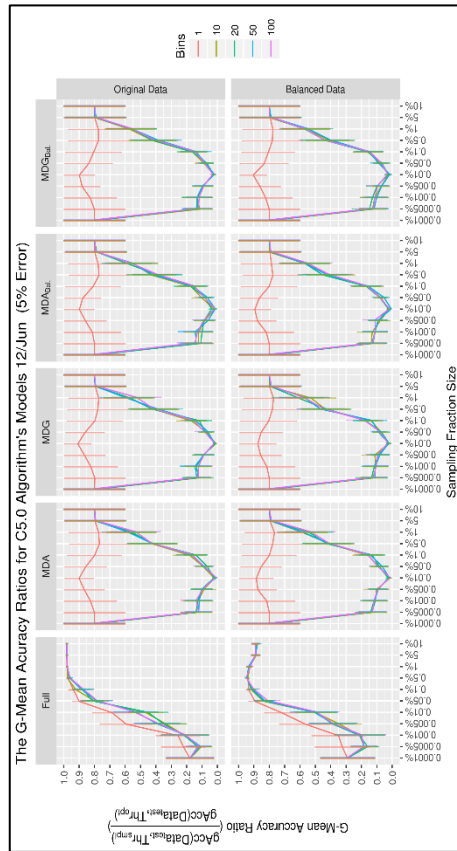
(a)



(b)

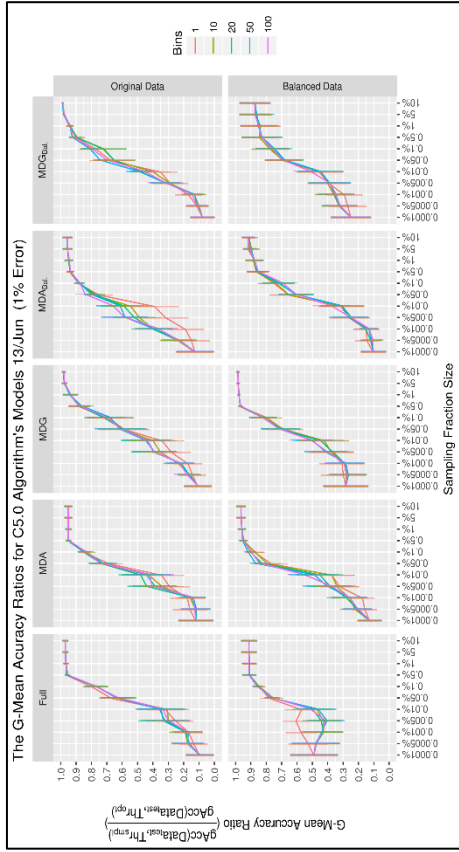


(c)

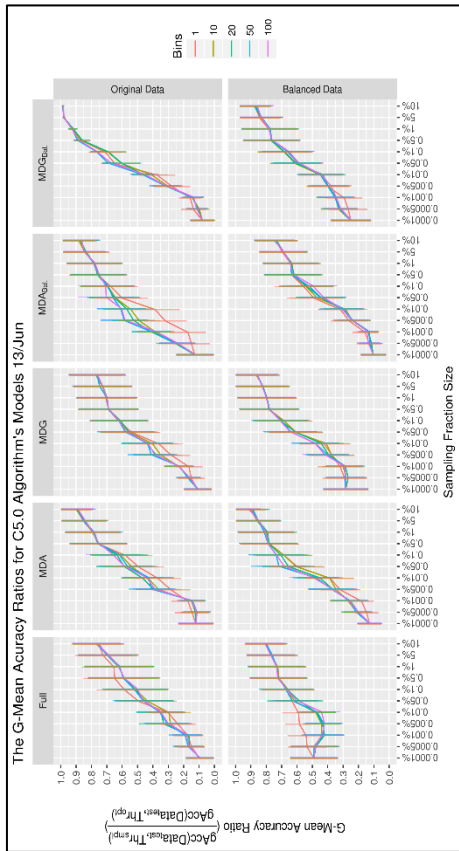


(d)

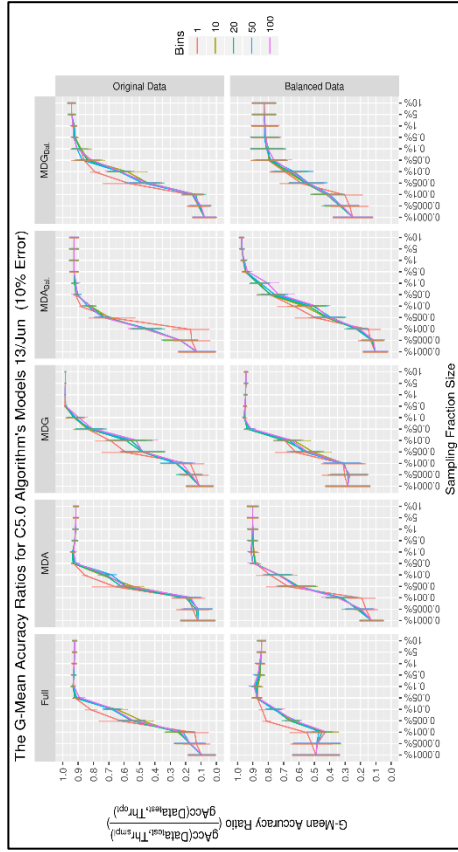
Figure C.1: Day 2 (12/Jun) results for the C5.0 models. (a) Plots of G-Mean Accuracy Ratio (GAR) of Day 6 models at an error rate of 1%. (b) GAR plots at an error rate of 10%. (c) GAR plots at an error rate of 5%. (d) GAR plots at an error rate of 0%.



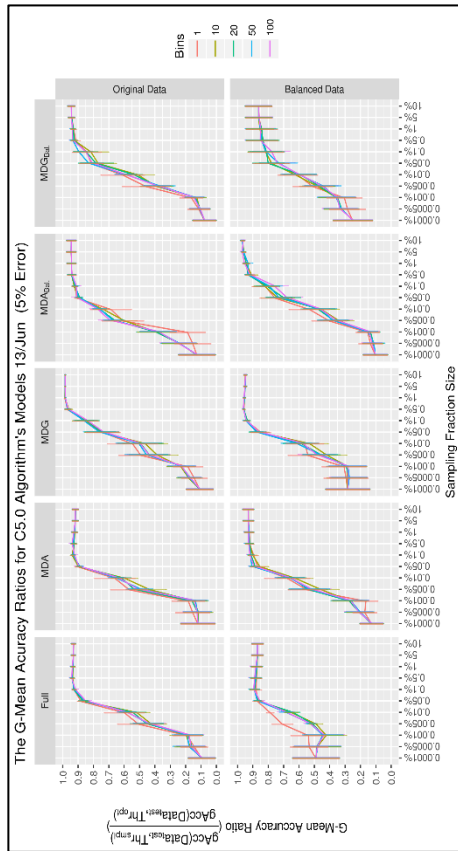
(a)



(b)

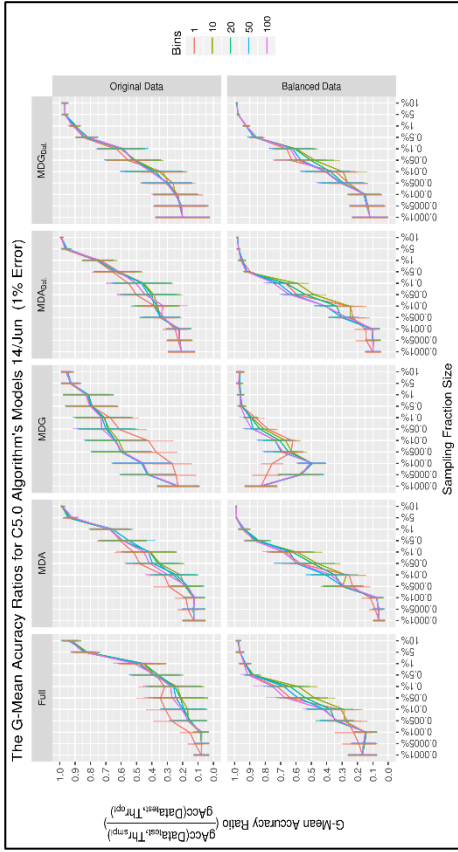


(c)

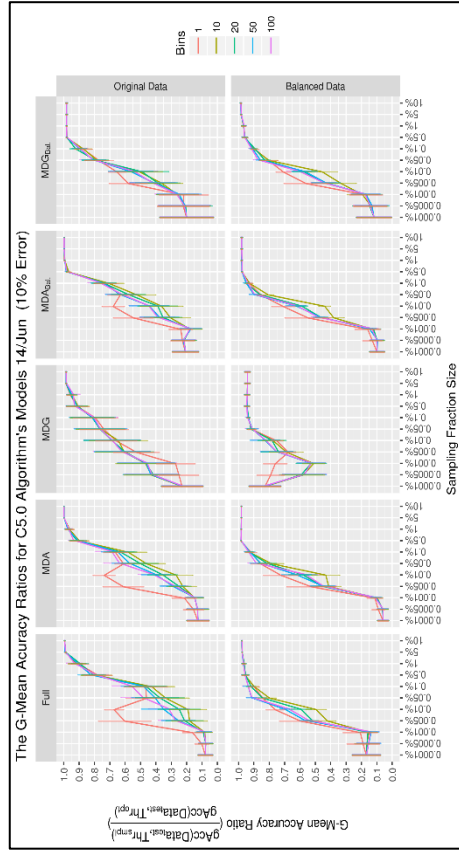


(d)

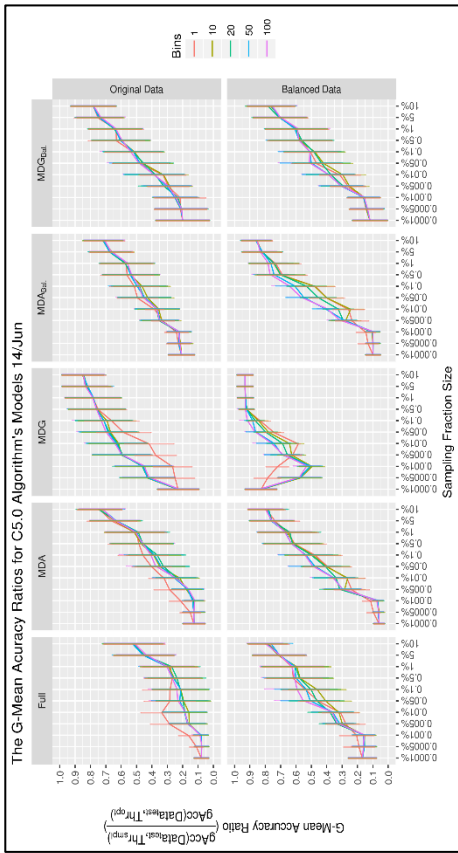
Figure C.2: Day 3 (13/Jun) results for the CS.0 models. (a) Plots of G-Mean Accuracy Ratio (GAR) of Day 6 models at an error rate of 1%. (b) GAR plots at an error rate of 5%. (c) GAR plots at an error rate of 10%. (d) GAR plots at an error rate of 20%.



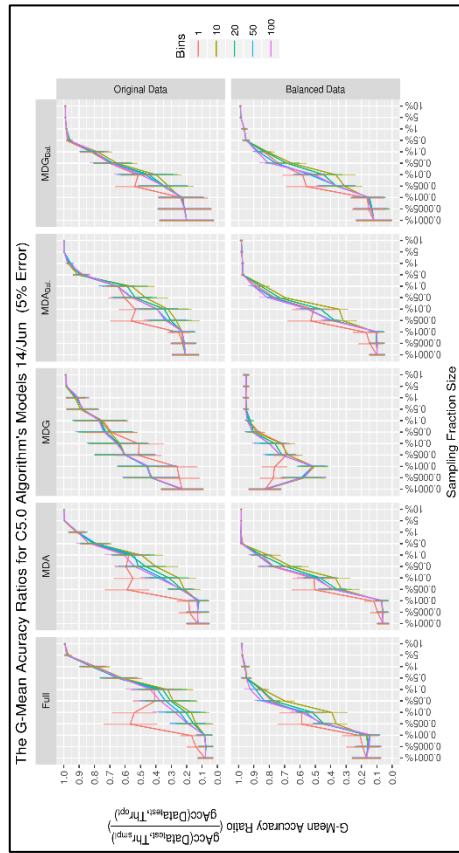
(a)



(b)

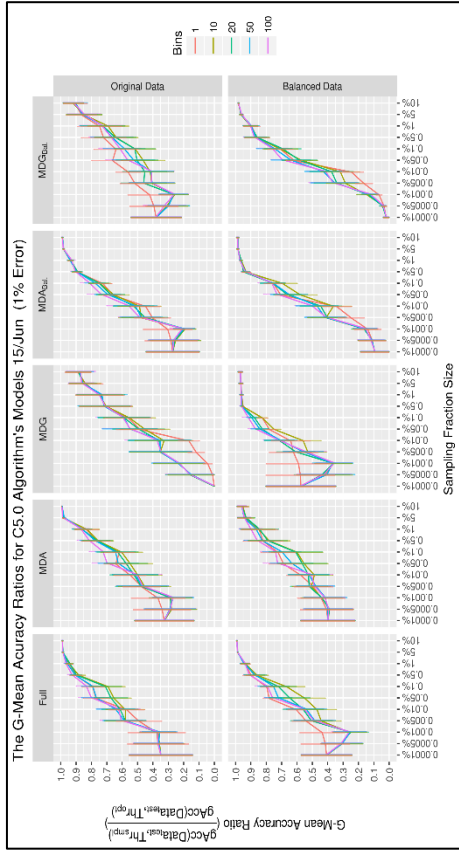


(c)

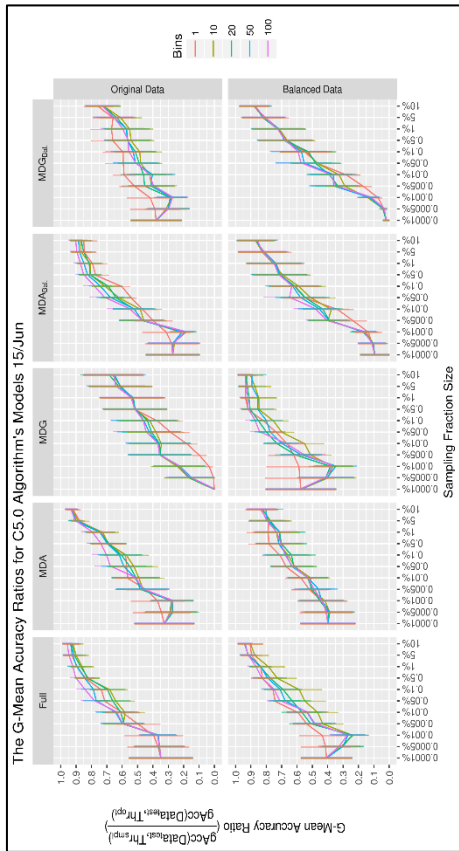


(d)

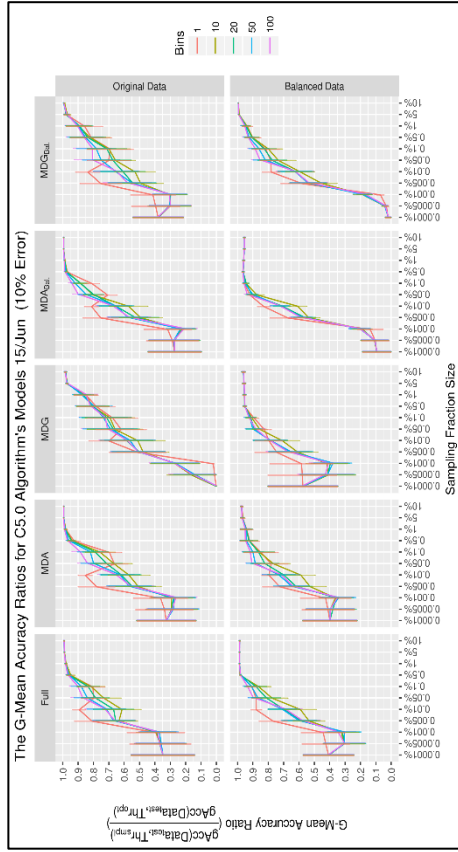
Figure C.3: Day 4 (14/Jun) results for the C5.0 models. (a) Plots of G-Mean Accuracy Ratio (GAR) of Day 6 models at an error rate of 1%. (b) GAR plots at an error rate of 10%. (c) GAR plots at an error rate of 5%. (d) GAR plots at an error rate of 0%.



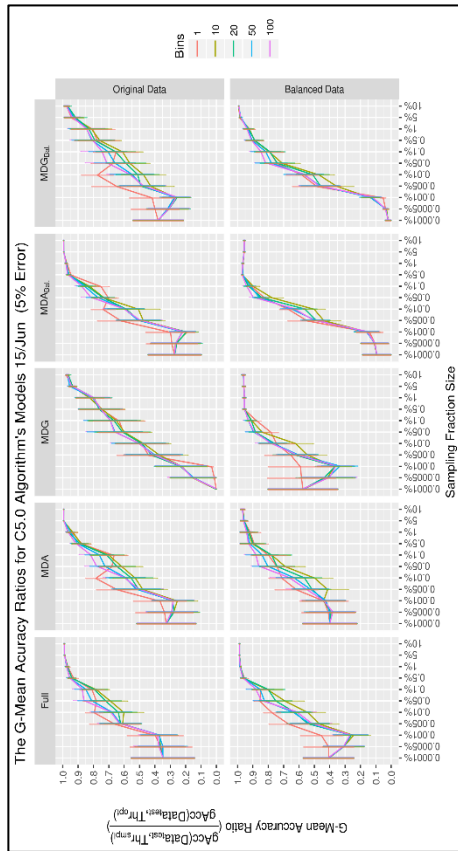
(a)



(b)

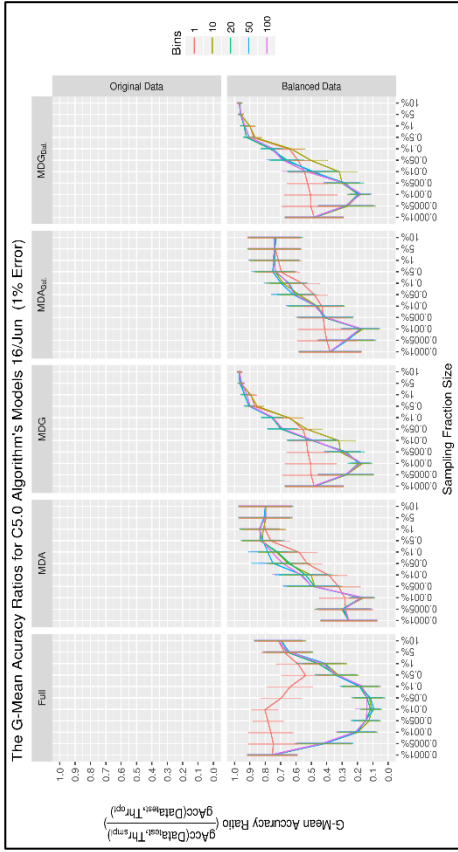


(c)

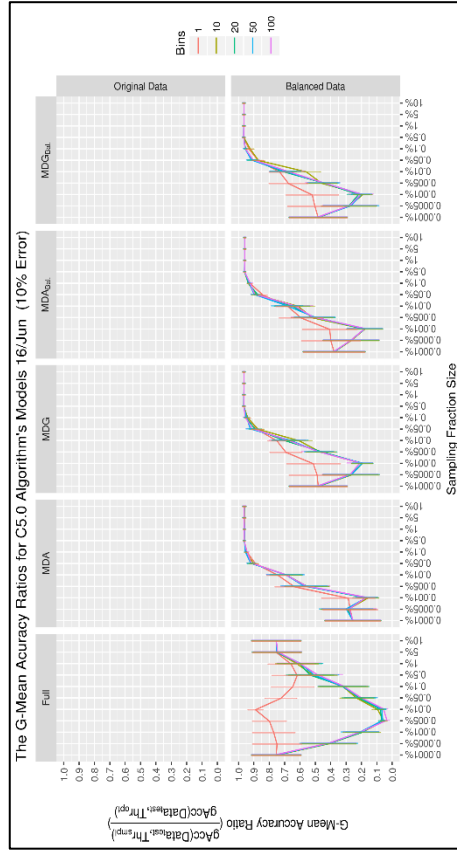


(d)

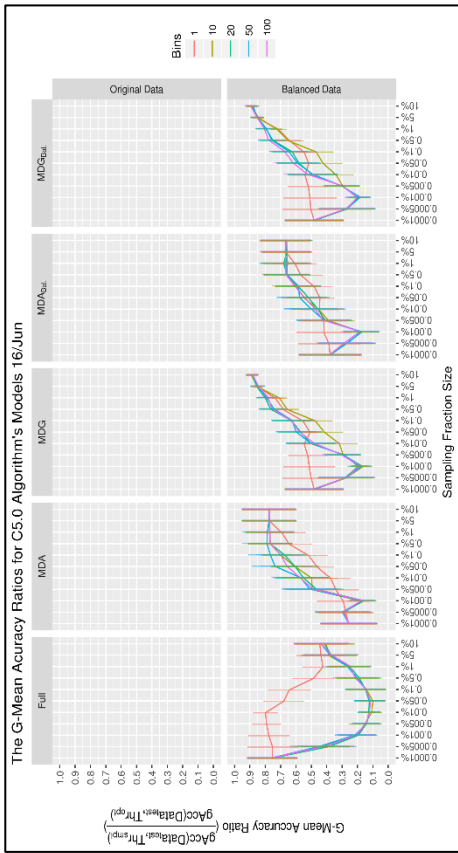
Figure C.4: Day 5 (15/Jun) results for the CS5.0 models. (a) Plots of G-Mean Accuracy Ratio (GAR) of Day 6 models at an error rate of 1%. (b) GAR plots at an error rate of 5%. (c) GAR plots at an error rate of 10%. (d) GAR plots at an error rate of 0%.



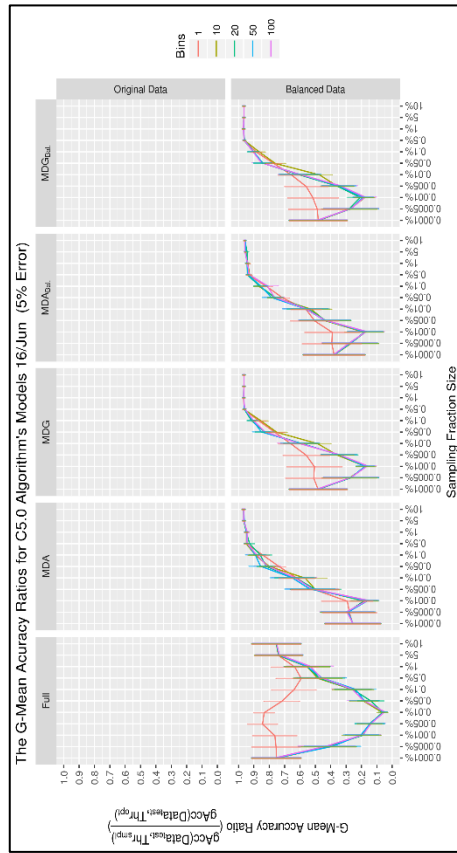
(a)



(b)

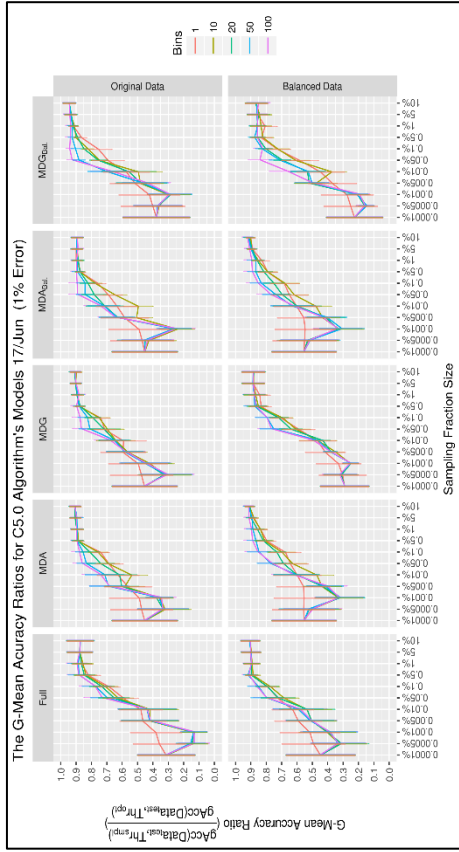


(c)

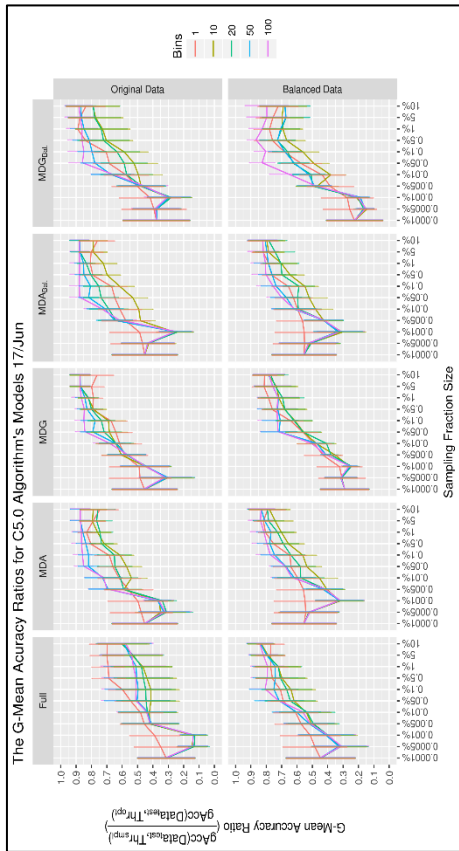


(d)

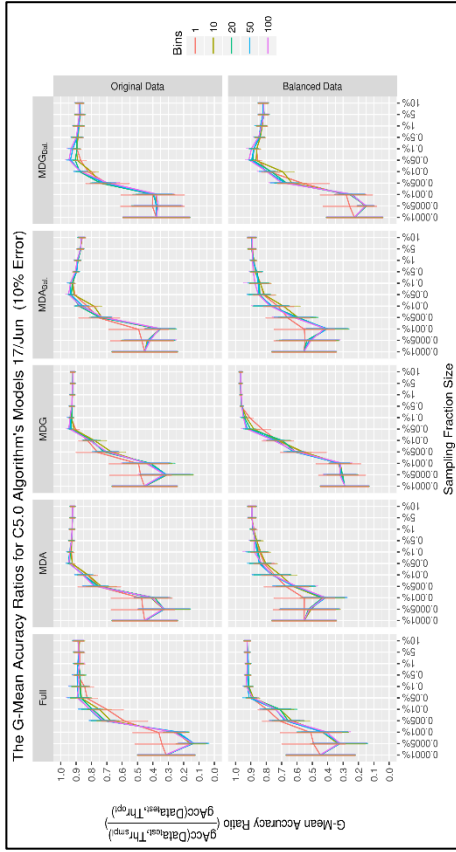
Figure C.5: Day 6 (16/Jun) results for the C5.0 models. (a) Plots of G-Mean Accuracy Ratio (GAR) of Day 6 models at an error rate of 1%. (b) GAR plots at an error rate of 10%. (c) GAR plots at an error rate of 5%. (d) GAR plots at an error rate of 10%.



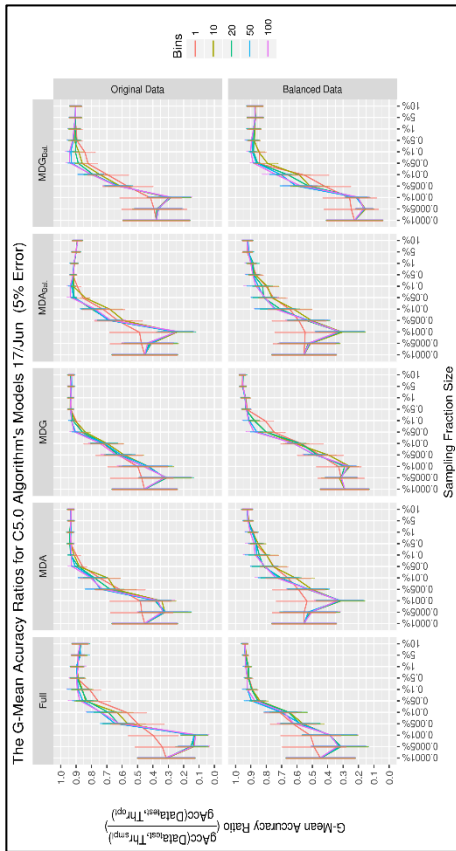
(a)



(b)

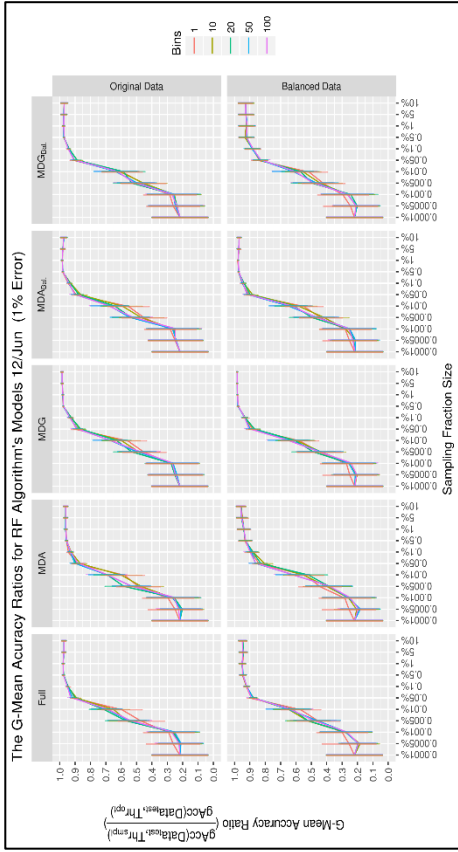


(c)

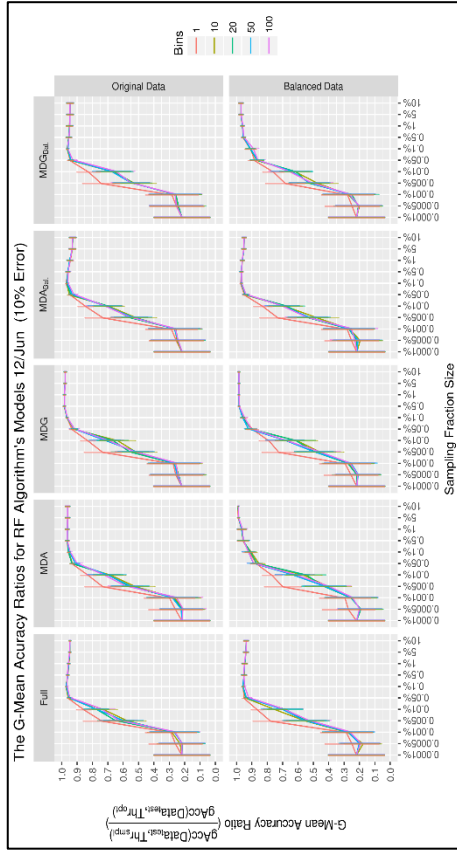


(d)

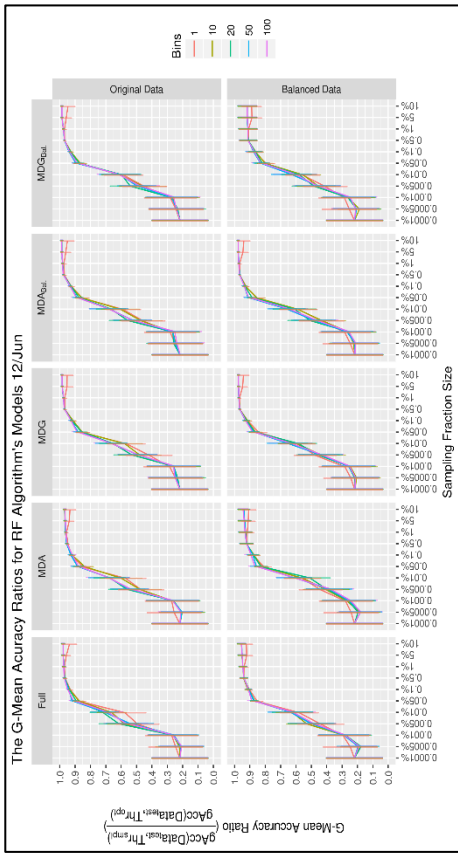
Figure C.6: Day 7 (17/Jun) results for the CS.0 models. (a) Plots of G-Mean Accuracy Ratio (GAR) of Day 6 models at an error rate of 1%. (b) GAR plots at an error rate of 5%. (c) GAR plots at an error rate of 10%. (d) GAR plots at an error rate of 20%.



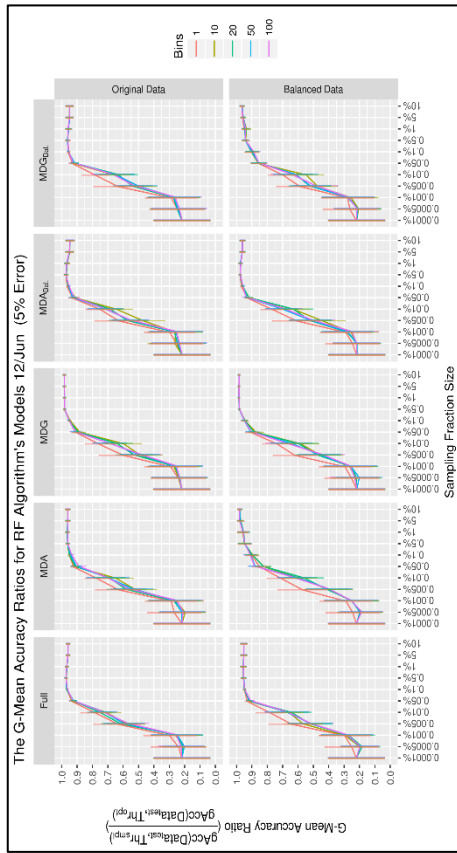
(b)



(d)

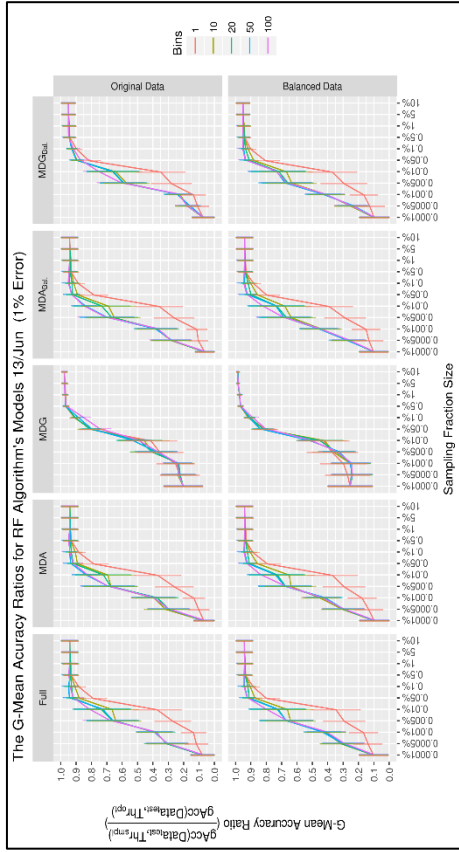


(a)

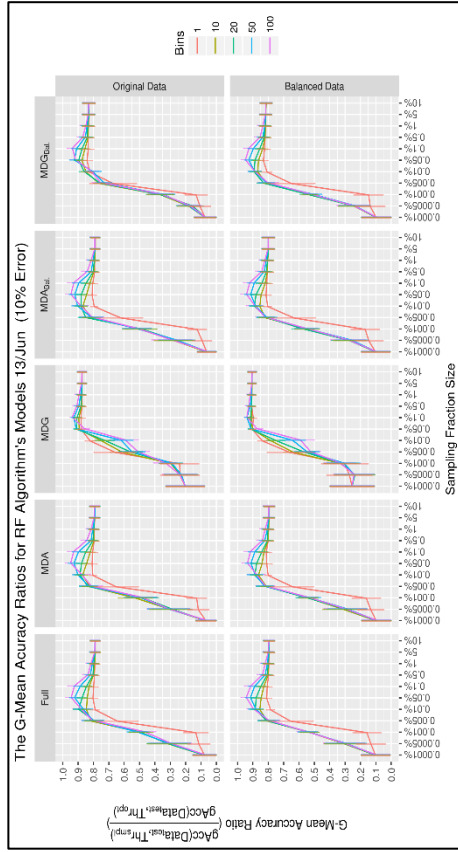


(c)

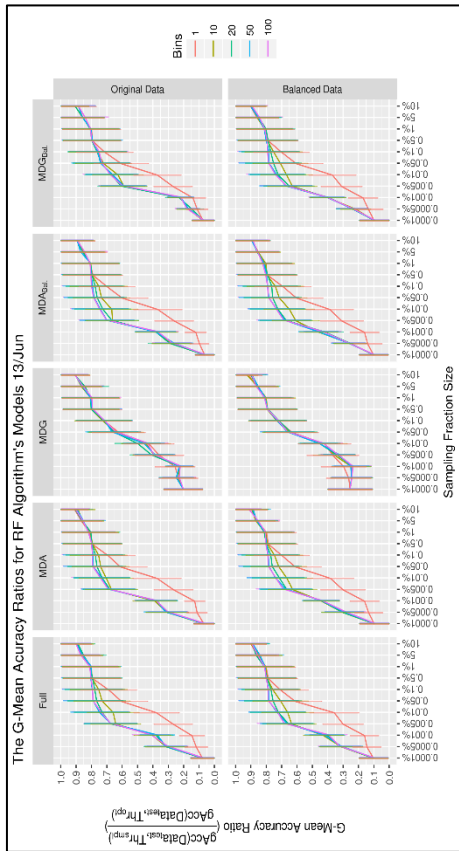
Figure C.7: Day 2 (12/Jan) results for the RF models. (a) Plots of G-Mean Accuracy Ratio (GAR) of Day 6 models at an error rate of 1%. (b) GAR plots at an error rate of 10%. (c) GAR plots at an error rate of 5%. (d) GAR plots at an error rate of 10%.



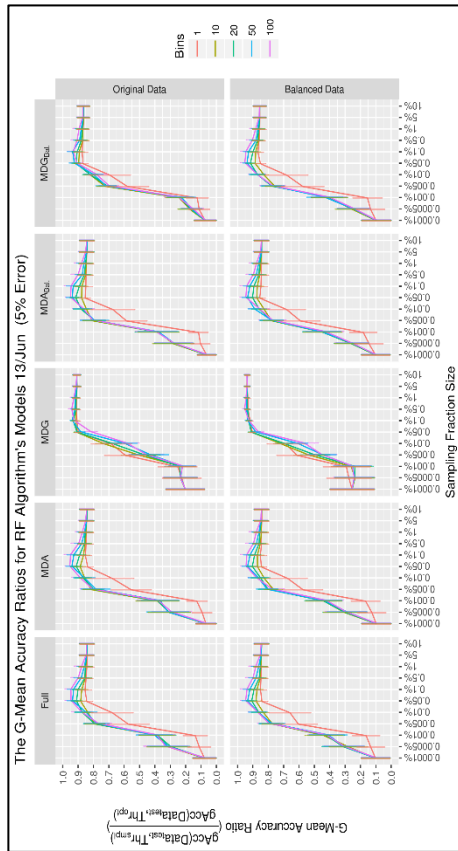
(b)



(d)

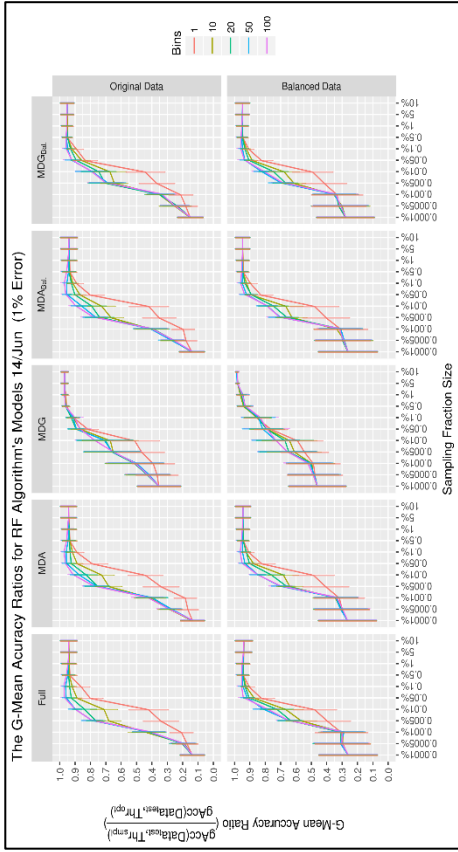


(a)

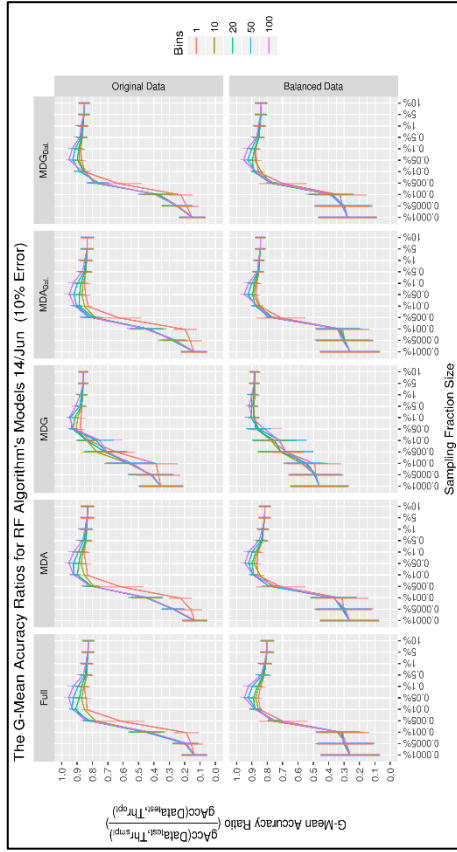


(c)

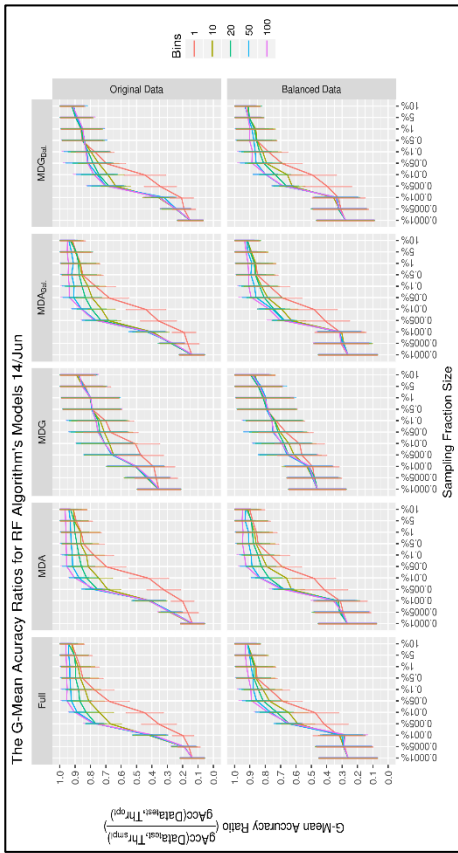
Figure C.8: Day 3 (13/Jun) results for the RF models. (a) Plots of G-Mean Accuracy Ratio (GAR) of Day 6 models at an error rate of 1%. (b) GAR plots at an error rate of 10%. (c) GAR plots at an error rate of 5%. (d) GAR plots at an error rate of 10%.



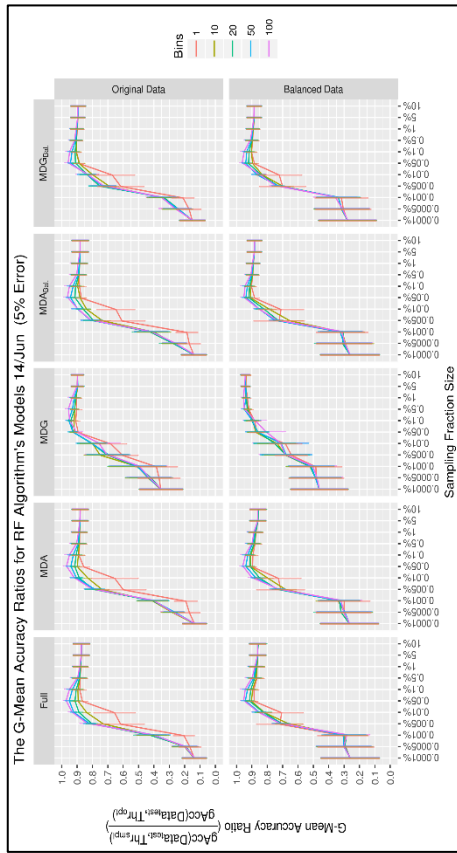
(a)



(b)

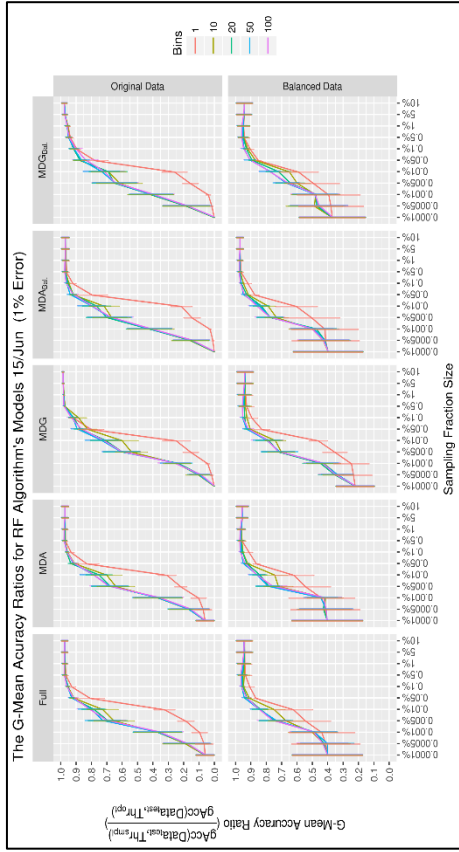


(c)

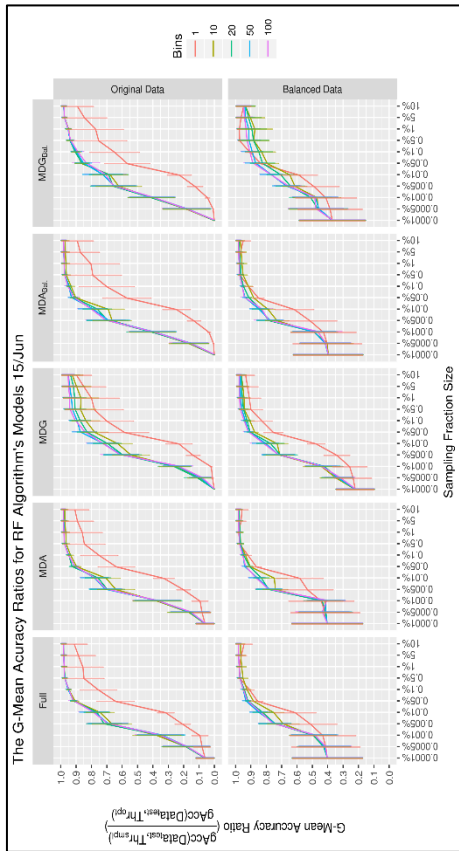


(d)

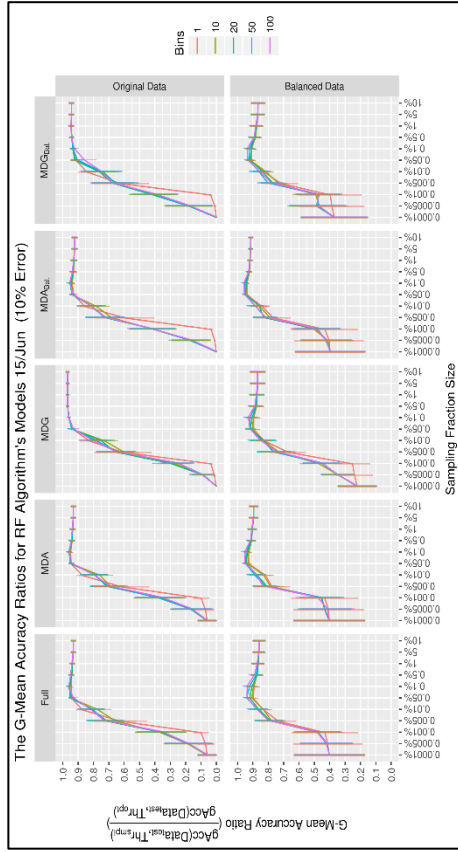
Figure C.9: Day 4 (14/June) results for the RF models. (a) Plots of G-Mean Accuracy Ratio (GAR) of Day 6 models at an error rate of 1%. (b) GAR plots at an error rate of 10%. (c) GAR plots at an error rate of 5%. (d) GAR plots at an error rate of 10%.



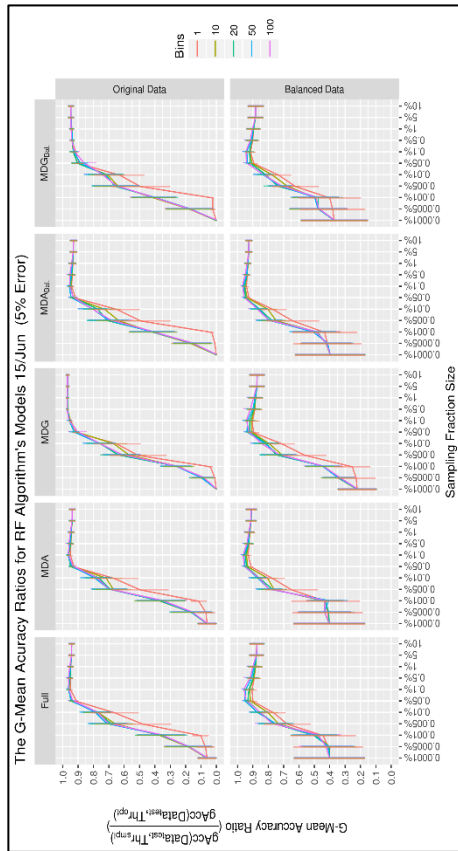
(a)



(b)

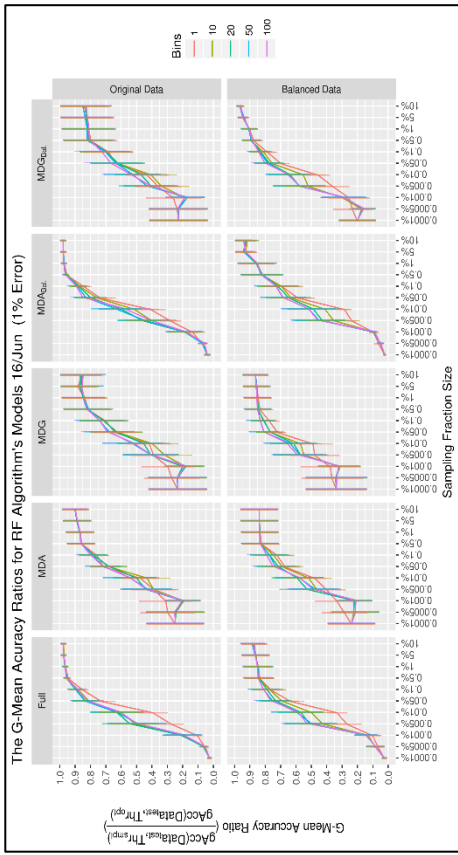


(c)

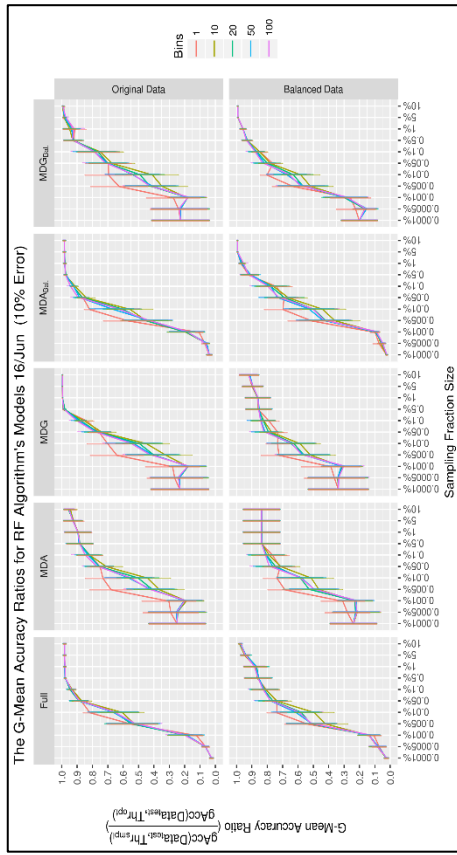


(d)

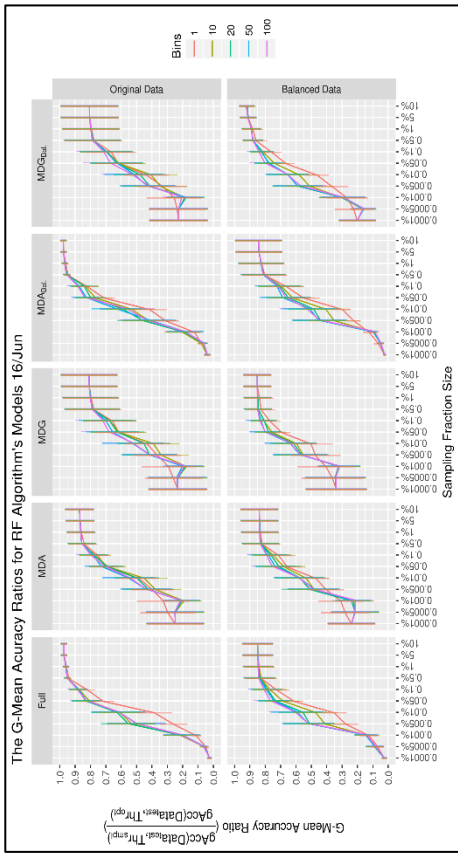
Figure C.10: Day 5 (15/Jun) results for the RF models. (a) Plots of G-Mean Accuracy Ratio (GAR) of Day 6 models at an error rate of 1%. (b) GAR plots at an error rate of 5%. (c) GAR plots at an error rate of 10%. (d) GAR plots at an error rate of 20%.



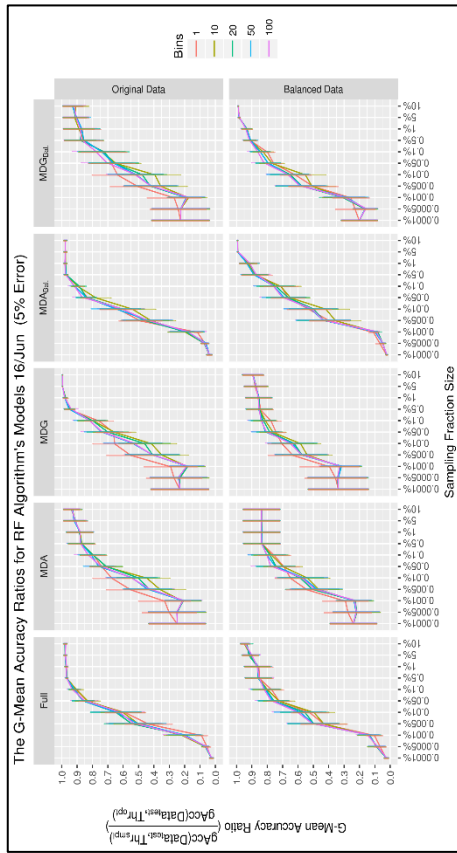
(a)



(b)

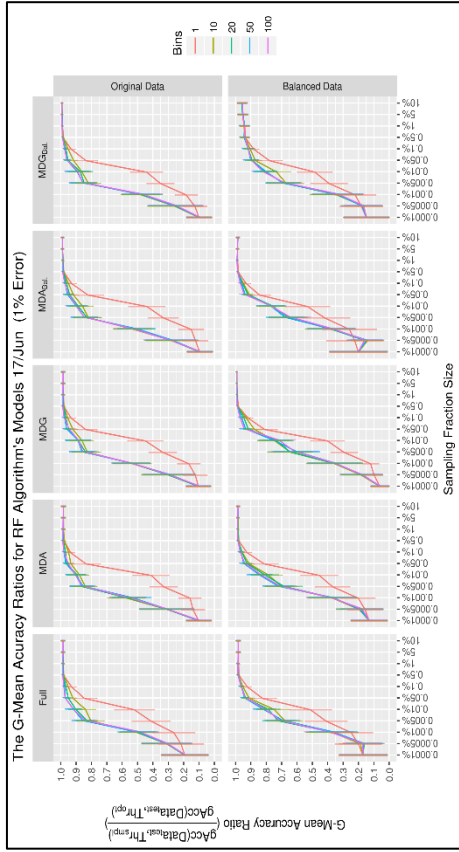


(c)

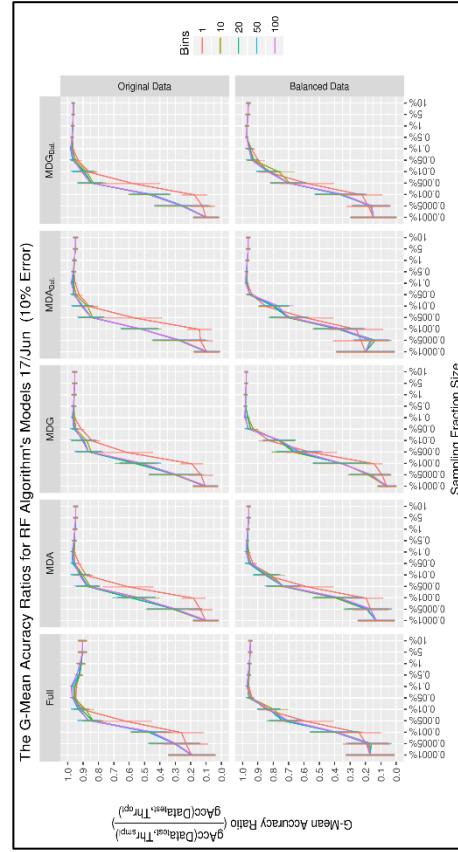


(d)

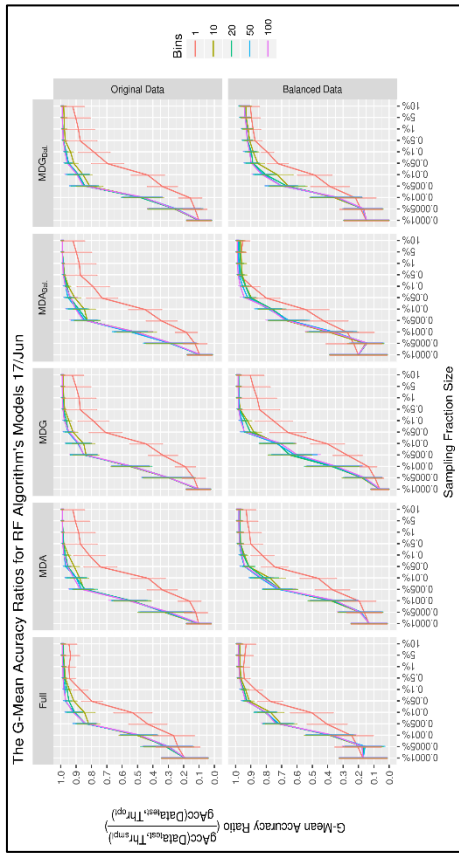
Figure C.11: Day 6 (16/June) results for the RF models. (a) Plots of G-Mean Accuracy Ratio (GAR) of Day 6 models at an error rate of 1%. (b) GAR plots at an error rate of 10%. (c) GAR plots at an error rate of 5%. (d) GAR plots at an error rate of 0%.



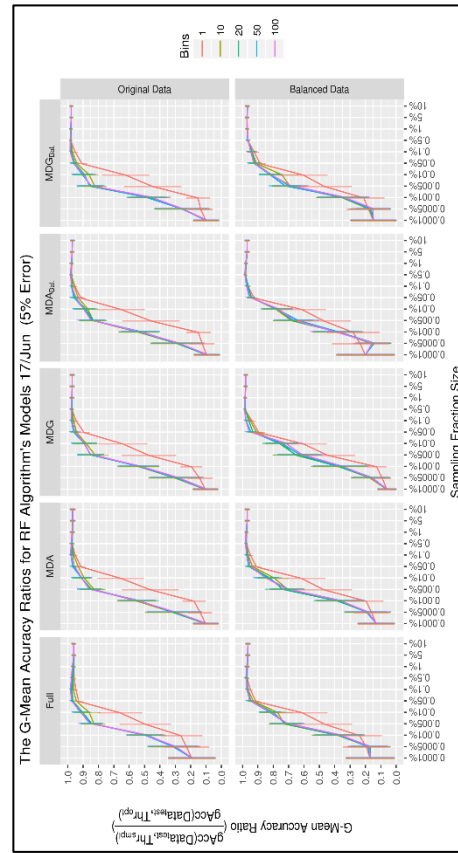
(b)



(d)

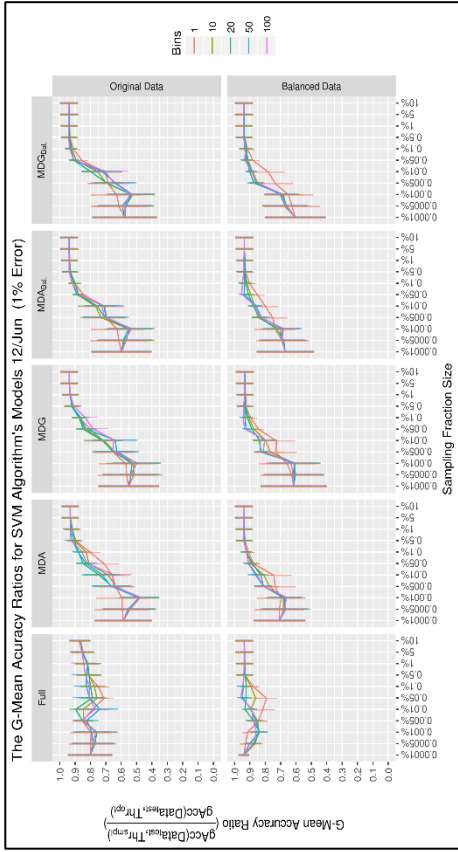


(a)

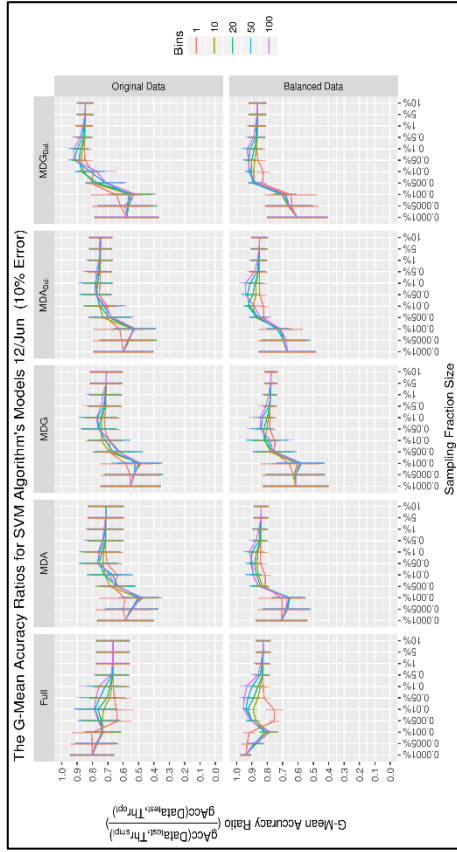


(c)

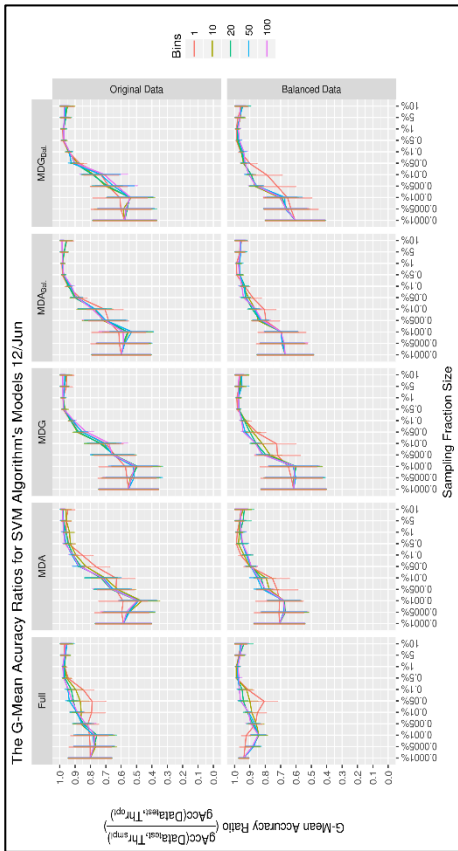
Figure C.12: Day 7 (17/Julun) results for the RF models. (a) Plots of G-Mean Accuracy Ratio (GAR) of Day 6 models at an error rate of 1%. (b) GAR plots at an error rate of 10%. (c) GAR plots at an error rate of 5%. (d) GAR plots at an error rate of 10%.



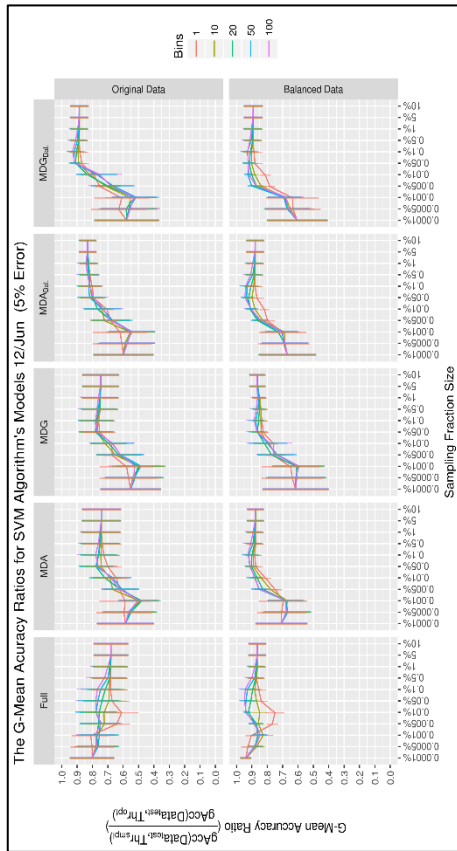
(a)



(b)

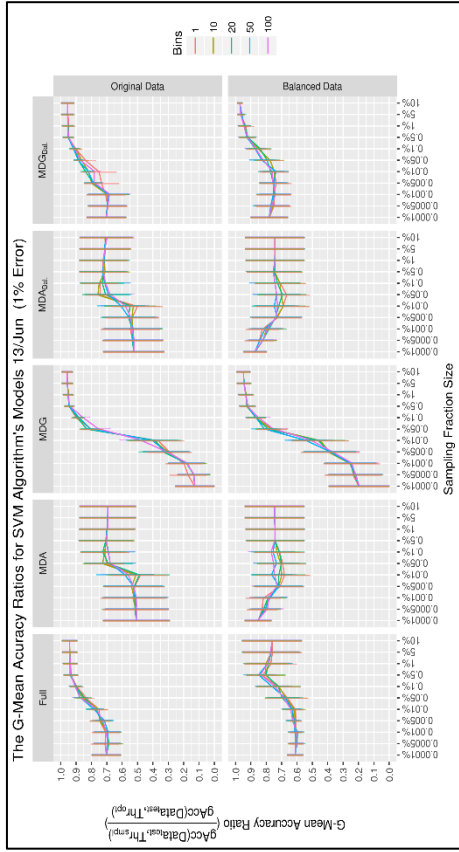


(c)

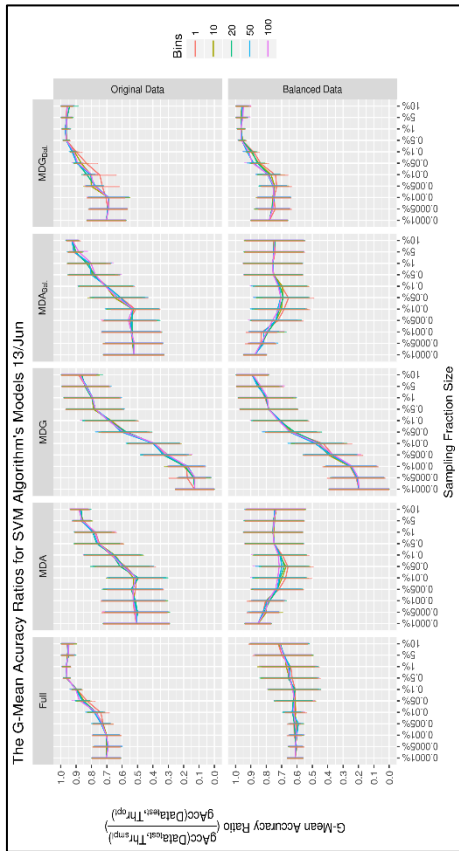


(d)

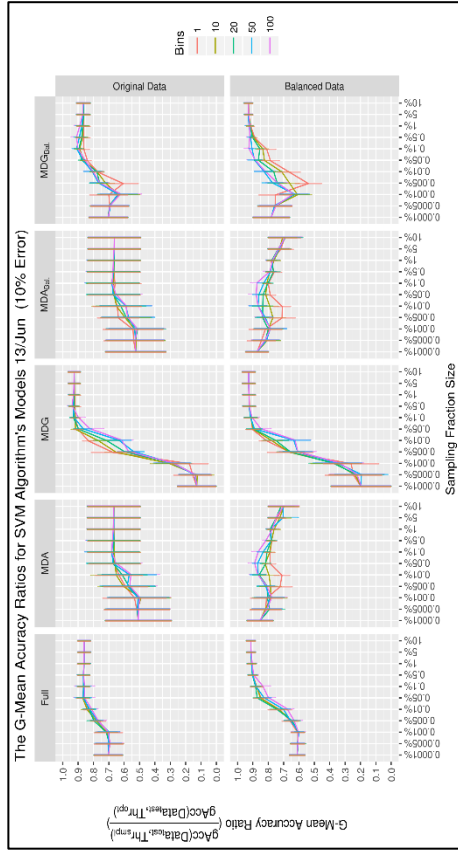
Figure C.13: Day 2 (12/Jan) results for the SVM models. (a) Plots of G-Mean Accuracy Ratio (GAR) of Day 6 models at an error rate of 1%. (b) GAR plots at an error rate of 10%. (c) GAR plots at an error rate of 5%. (d) GAR plots at an error rate of 10%.



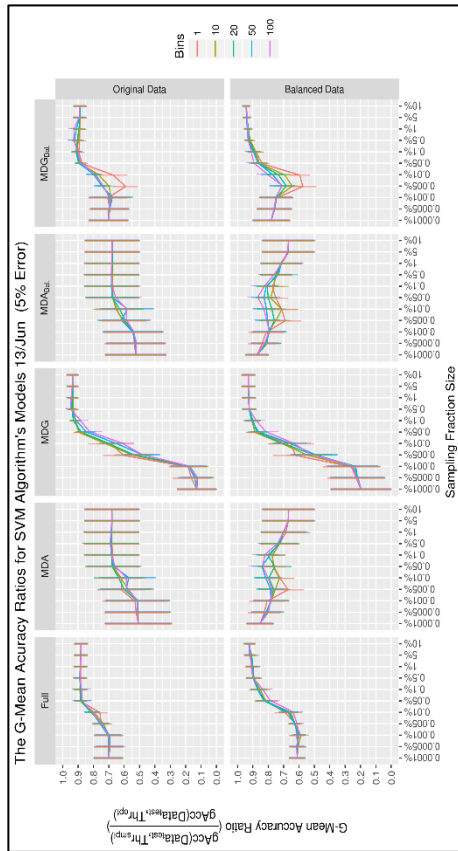
(a)



(b)

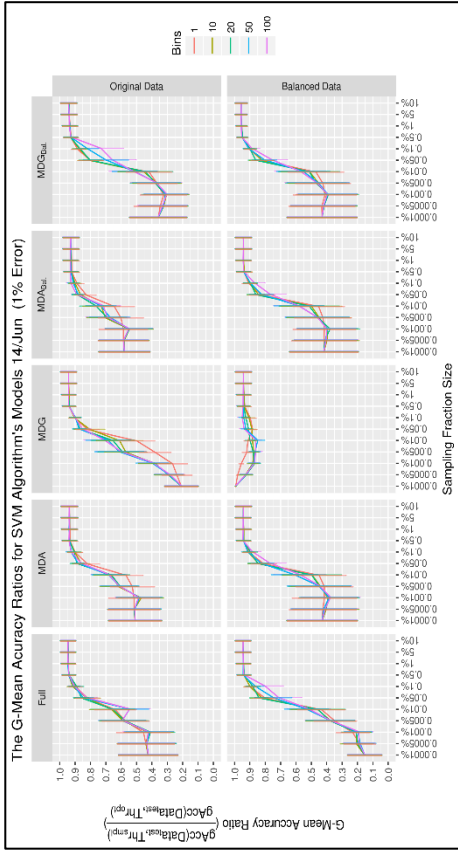


(c)

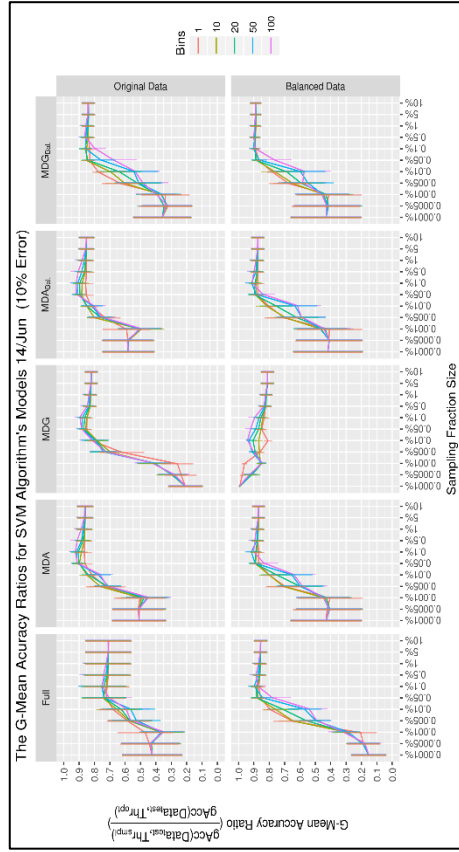


(d)

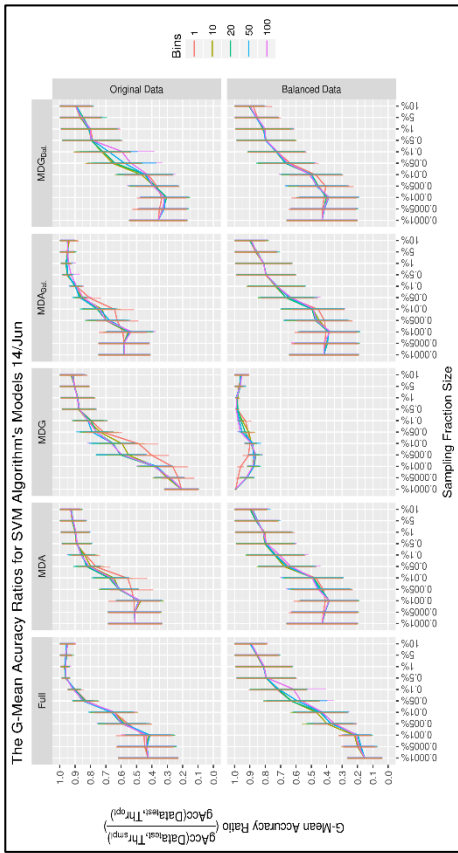
Figure C.14: Day 3 (13/Jun) results for the SVM models. (a) Plots of G-Mean Accuracy Ratio (GAR) of Day 6 models at an error rate of 1%. (b) GAR plots at an error rate of 5%. (c) GAR plots at an error rate of 10%. (d) GAR plots at an error rate of 1%.



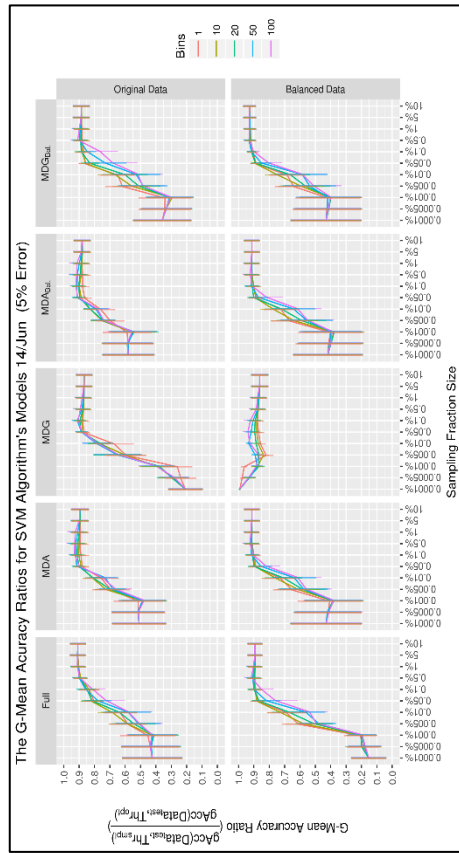
(a)



(b)

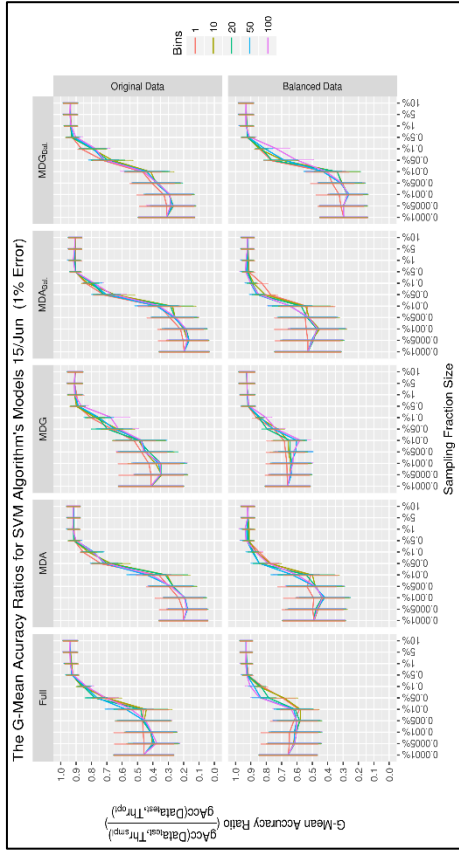


(c)

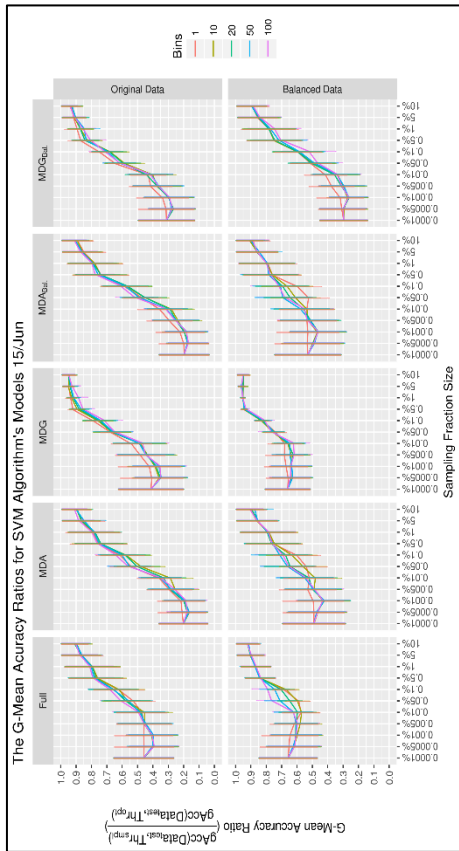


(d)

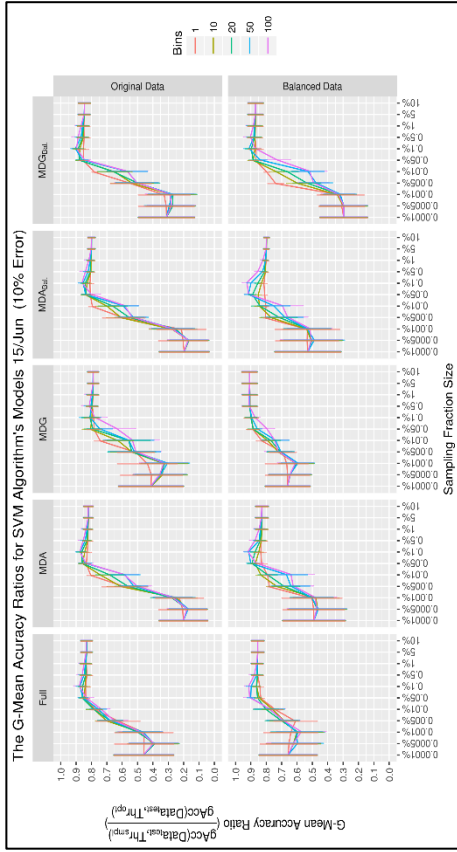
Figure C.15: Day 4 (14/Jun) results for the SVM models. (a) Plots of G-Mean Accuracy Ratio (GAR) of Day 6 models at an error rate of 1%. (b) GAR plots at an error rate of 10%. (c) GAR plots at an error rate of 5%. (d) GAR plots at an error rate of 0%.



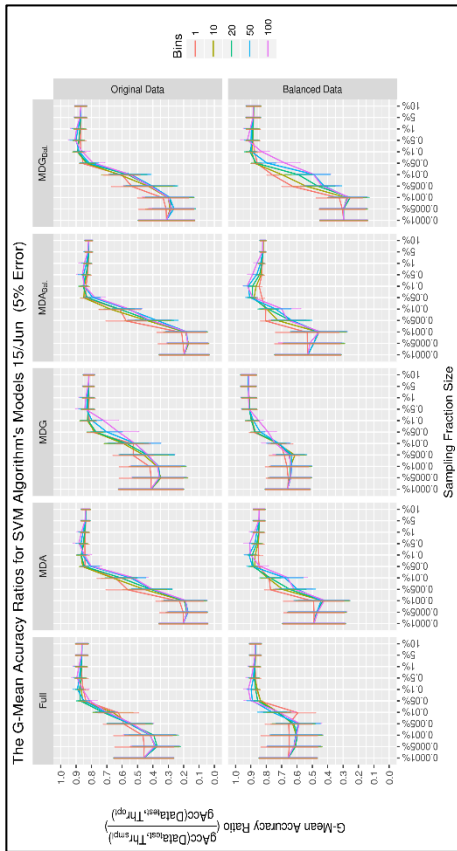
(a)



(b)

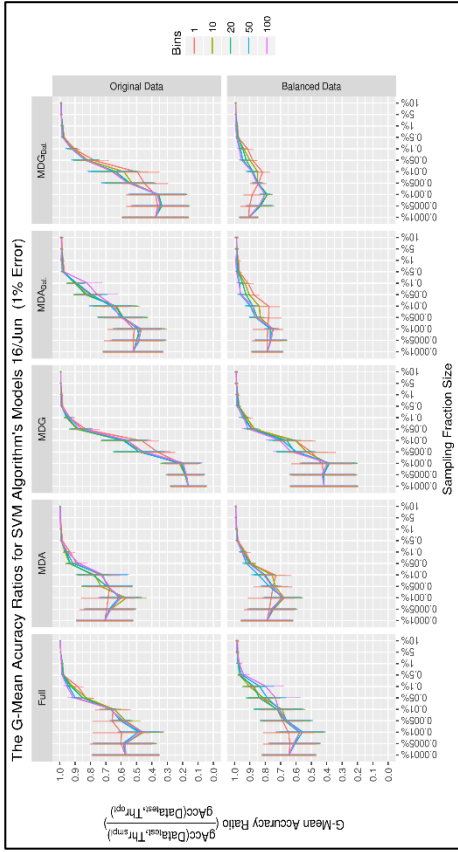


(c)

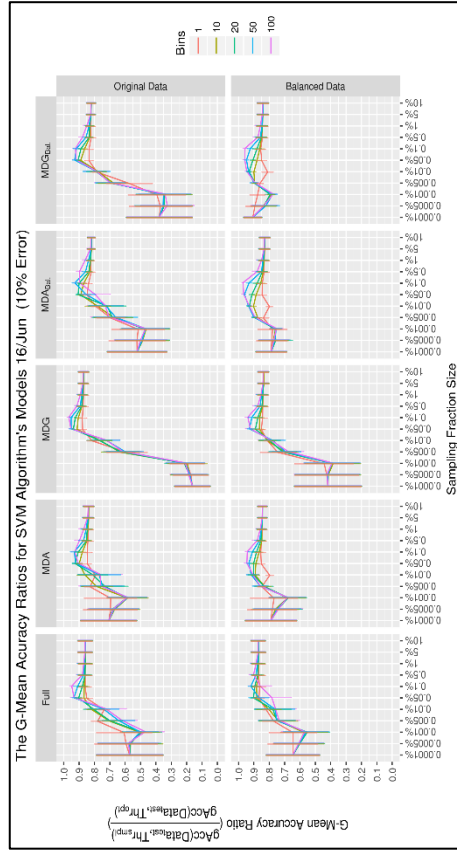


(d)

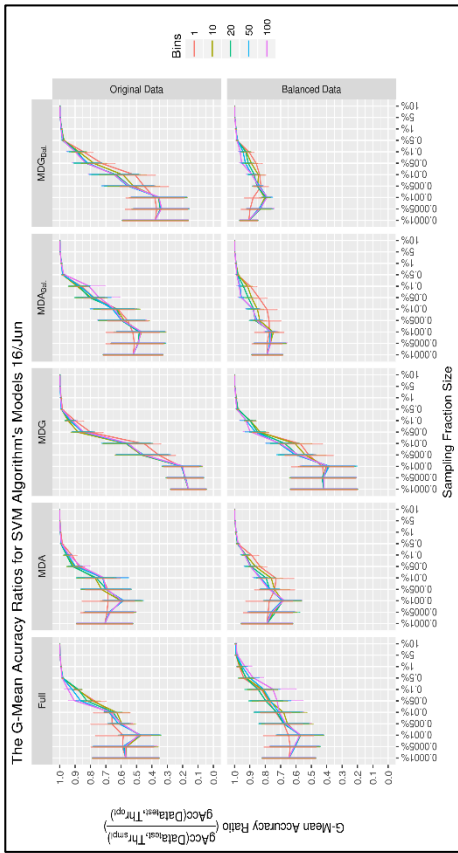
Figure C.16: Day 5 (15/Jun) results for the SVM models. (a) Plots of G-Mean Accuracy Ratio (GAR) of Day 6 models at an error rate of 1%. (b) GAR plots at an error rate of 5%. (c) GAR plots at an error rate of 10%. (d) GAR plots at an error rate of 1%.



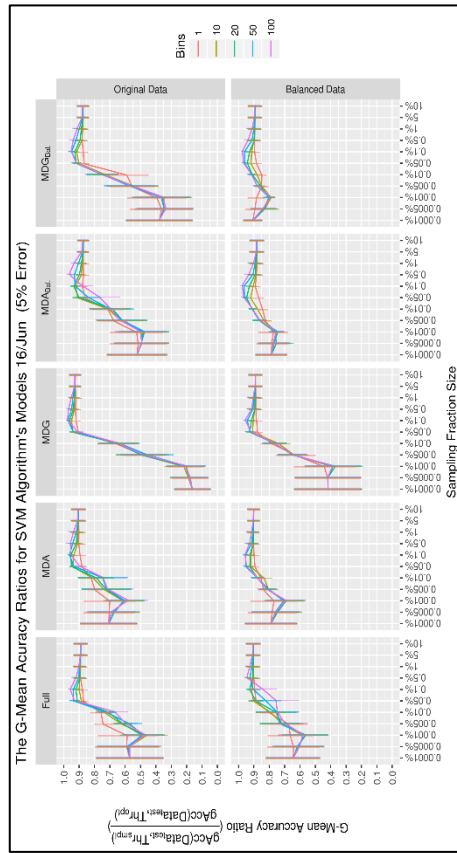
(a)



(b)

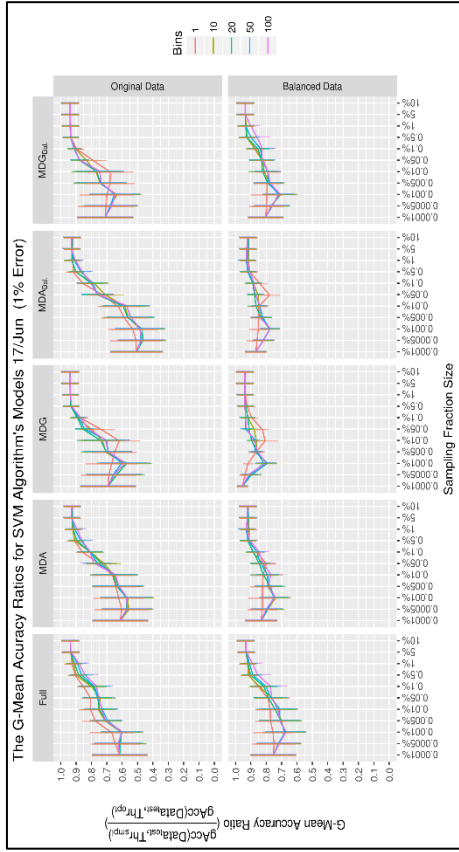


(c)

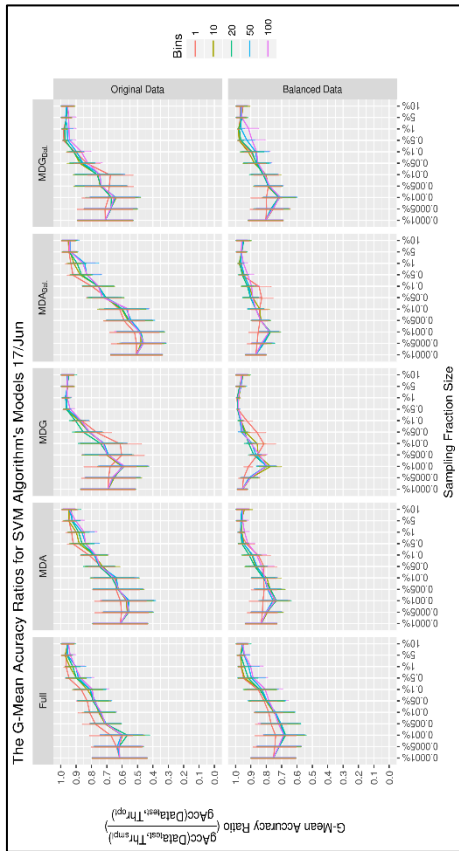


(d)

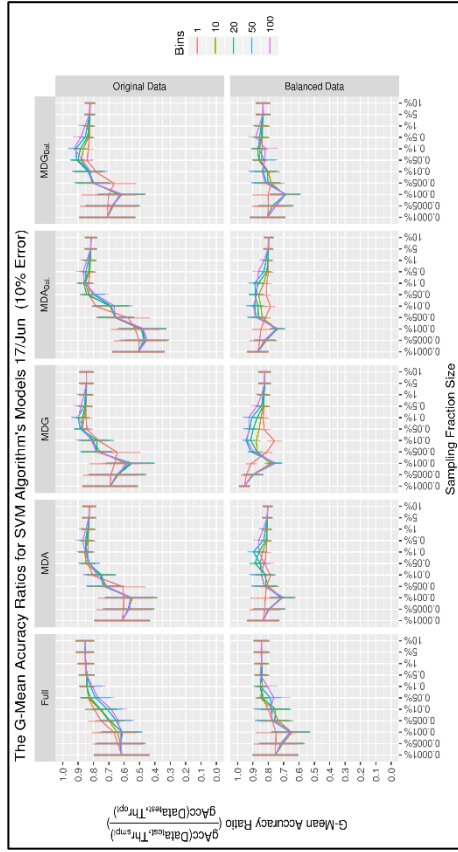
Figure C.17: Day 6 (16/Jun) results for the SVM models. (a) Plots of G-Mean Accuracy Ratio (GAR) of Day 6 models at an error rate of 1%. (b) GAR plots at an error rate of 10%. (c) GAR plots at an error rate of 5%. (d) GAR plots at an error rate of 10%.



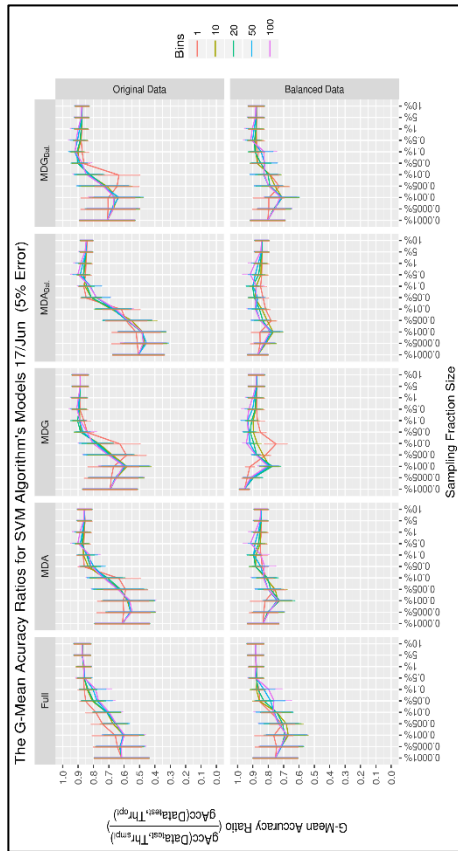
(a)



(b)



(c)



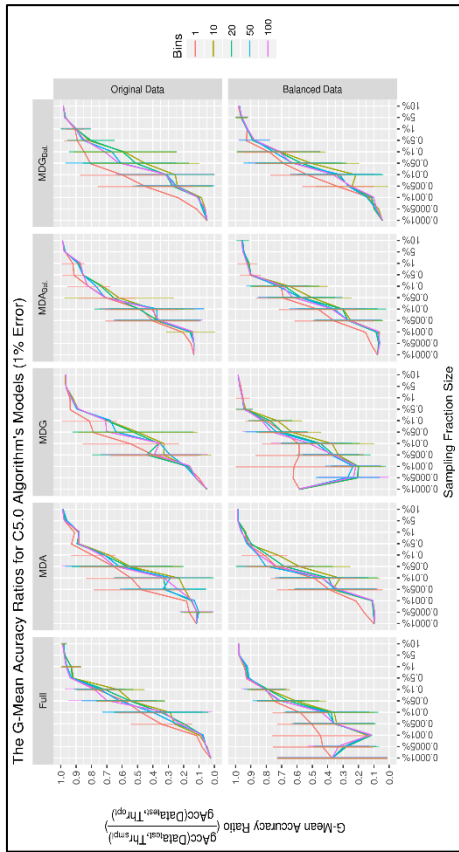
(d)

Figure C.18: Day 7 (17/Jul) results for the SVM models. (a) Plots of G-Mean Accuracy Ratio (GAR) of Day 6 models at an error rate of 1%. (b) GAR plots at an error rate of 10%. (c) GAR plots at an error rate of 5%. (d) GAR plots at an error rate of 0%.

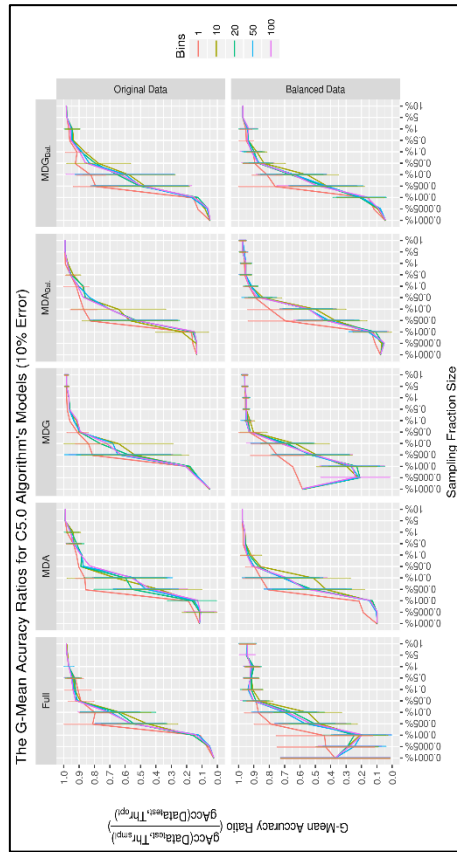
C.2. Models GAR Plots

The results of the experiments discussed in **Chapter 7** are illustrated in this section as G-Mean Accuracy Ratio (GAR) plots for every ML algorithm. Each one of the figures (**Figure C.19**, **Figure C.20** and **Figure C.21**) shows the results for one of the three ML algorithms (C5.0, RF and SVM) respectively. Each one of the plots (a, b, c and d) in every figure shows the results using different error rates (0%, 1%, 5% and 10%). Where each subplot shows the median of the GAR values for all of the models with the same feature set and data balance group for that algorithm. The curves in each subplot for every group illustrate the medians of the different sampling strategies (B_1 , B_{10} , B_{20} , B_{50} and B_{100}) for every sample size (10% to 0.0001%).

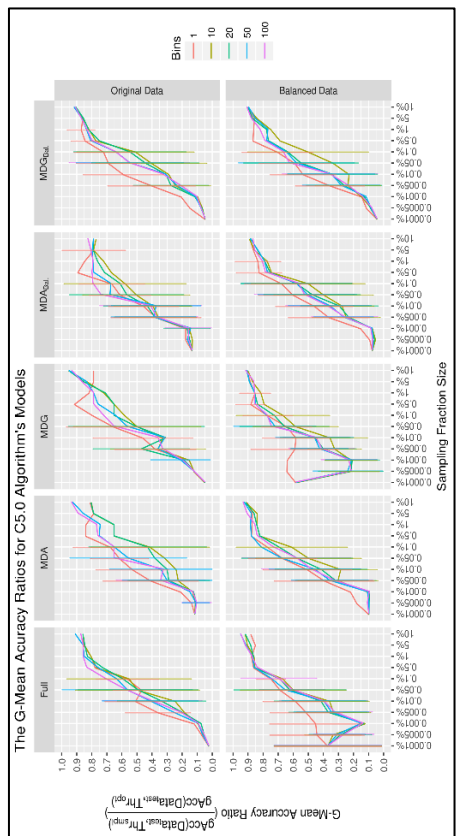
C.2.1. C5.0 (Decision Trees) Results



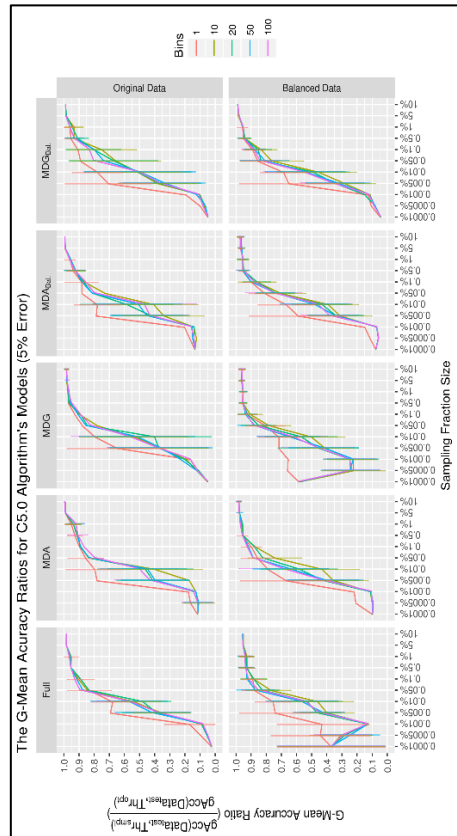
(b)



(d)



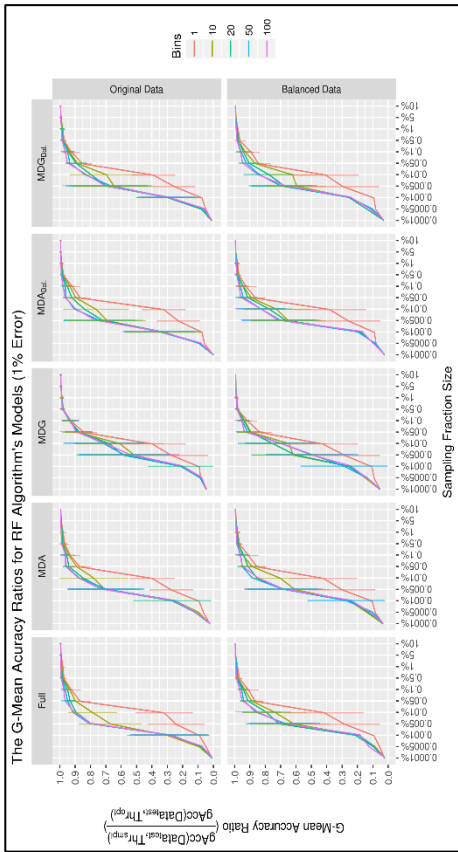
(a)



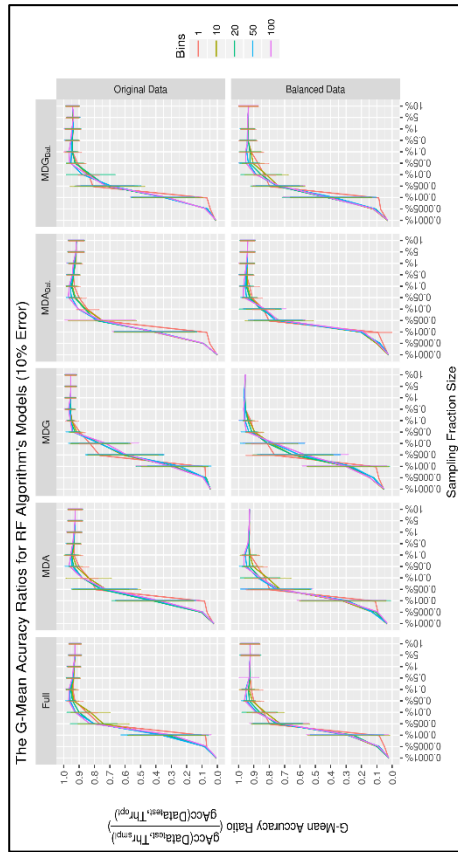
(c)

Figure C.19: Medians of G-Mean Accuracy Ratios (GAR) of C5.0 models for every feature set and data balance type combination. (a) Medians at an error rate of 0%. (b) Medians at error rate of 1%. (c) Medians at an error rate of 5%. (d) Medians at an error rate of 10%.

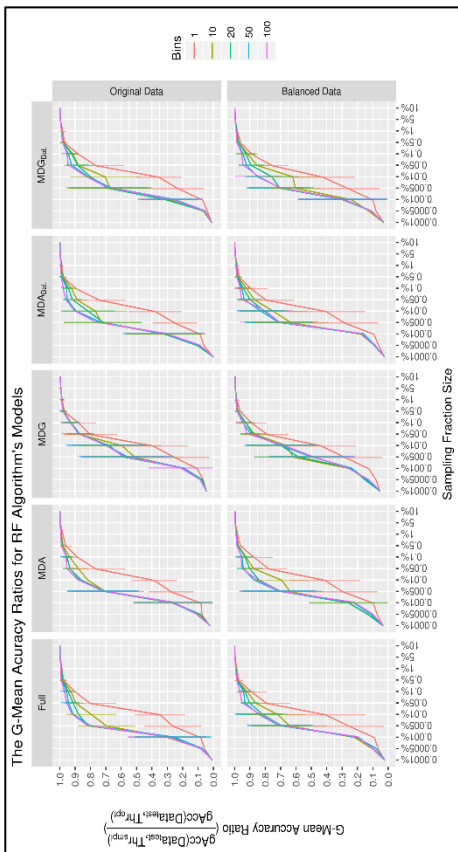
C.2.2. Random Forest (RF) Results



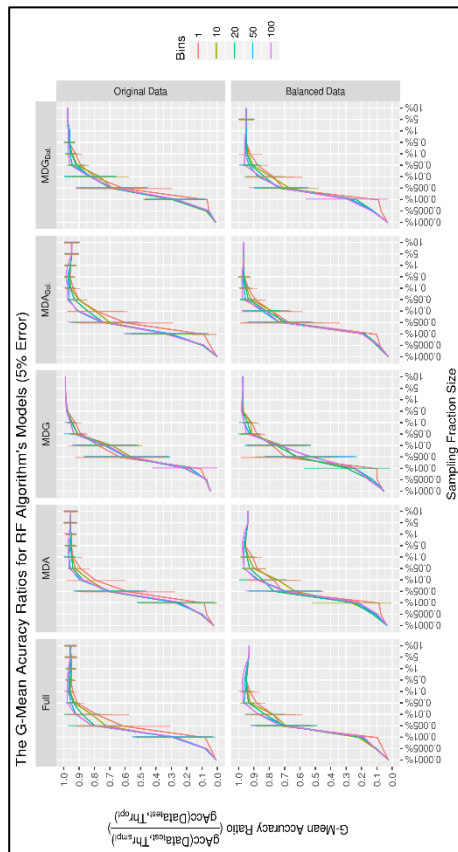
(a)



(b)



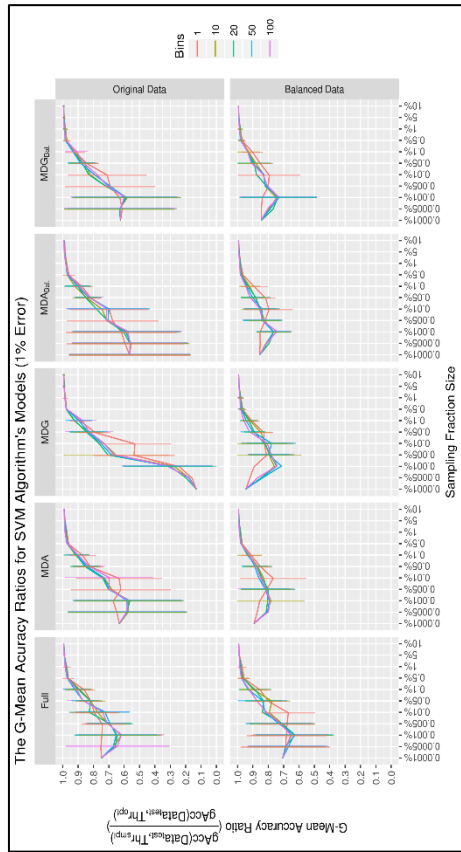
(c)



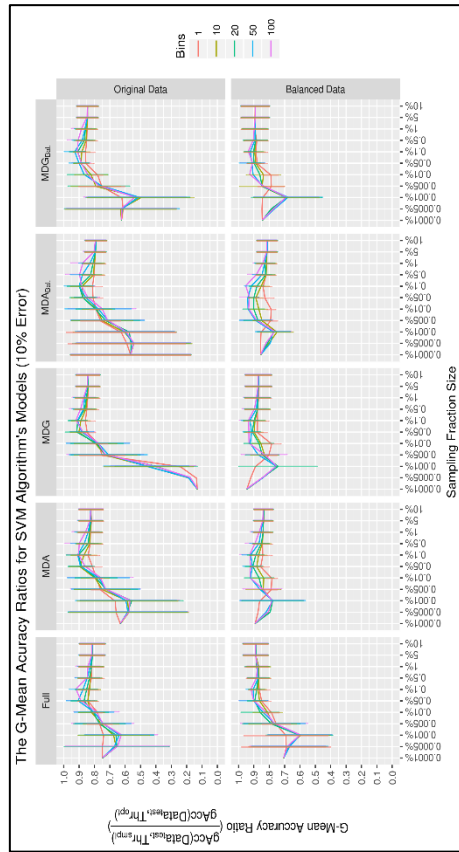
(d)

Figure C.20: Medians of G-Mean Accuracy Ratios (GAR) of RF models for every feature set and data balance type combination. (a) Medians at an error rate of 0%. (b) Medians at error rate of 1%. (c) Medians at an error rate of 5%. (d) Medians at an error rate of 10%.

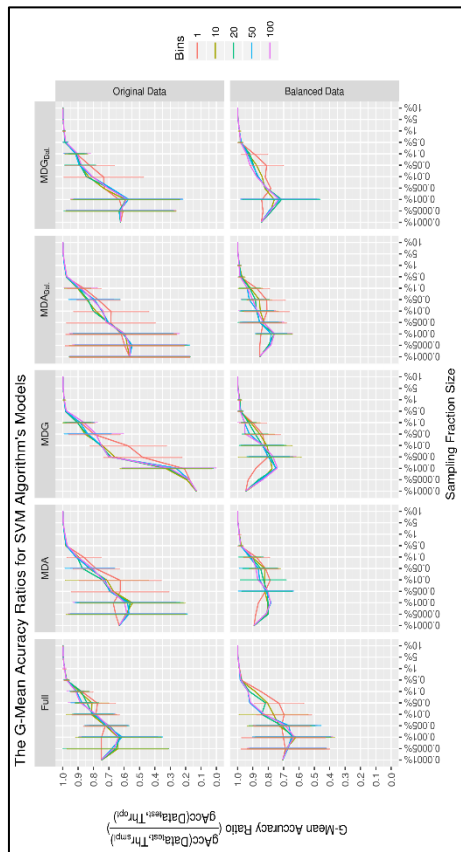
C.2.3. Support Vector Machine (SVM) Results



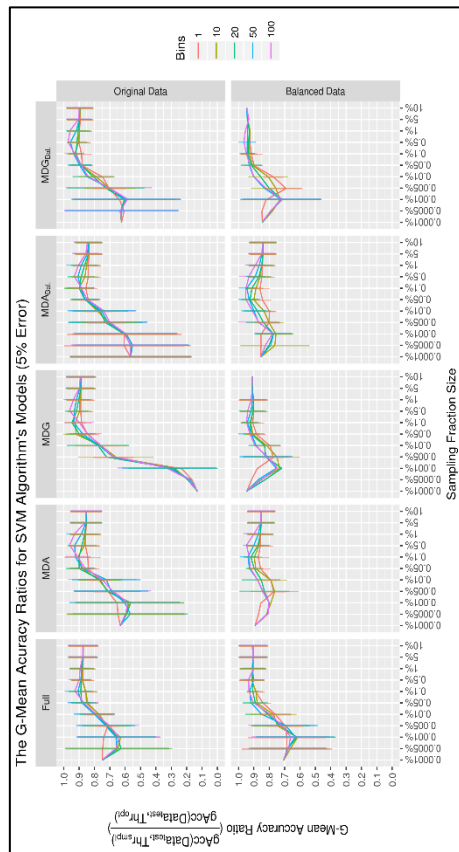
(a)



(b)



(c)



(d)

Figure C.21: Medians of G-Mean Accuracy Ratios (GAR) of SVM models for every features set and data balance type combination. (a) Medians at an error rate of 0%. (b) Medians at error rate of 1%. (c) Medians at an error rate of 5%. (d) Medians at an error rate of 10%.



Appendix (D) Feature Descriptions of STA2018 Dataset

This appendix lists the **638** features that were extracted from the UNB ISCX 2012 dataset and derived using Onut's schema [15]. A description is provided for each of the features. All features deleted at the clean-up phase are marked (in red italics) and their sequence number has been concatenated with a sequence of letters.

The following is a summary of the features generated and/or deleted at every transformation stage (they have been placed in order of number of the resultant dataset, i.e. *STA2018*):

- **Basic-features extraction:** a total of 193 features were extracted at this phase. These features had the sequences {1-2, 4-5, 7-12, 15-144}.
- **Validation and connection labelling:** this phase added one more variable which was the class feature and included the label {Normal or Attack} for every connection.
- **Extend:** this phase extended the feature space by deriving the following features groups (all of the descriptions of these extended features are taken from Onut's PhD thesis [385]):
 - *Connection-based features:* a total of 220 features were derived and start with "DFMC_*" (this code represented the path of the tree presented in **Figure 5.2** as coded by Onut [15, 385]). These features had the sequence {145-348}

- *Time-based features*: a total of **220** features were derived and start with “DFMT_*” (this code represents the path of the tree presented in **Figure 5.2** as coded by Onut [15, 385]). These features had the sequence {349-549}
- **Balance**: a total of **two** features (**synthetic** and **origOrder**) were added at this stage to distinguish the original connections from the synthetic connections. The sequences of these features are {13-14} and they were added to the basic features.
- **Clean up**: **two** features were added at this stage: **src_zone** and **dst_zone**. Their sequences are {3 and 6} and they were added to the basic features group. Also, a total of 88 useless features were removed at this stage: 53 from basic-features; 19 from the connection-based features group; and 16 from the time-based group.

D.1. Basic Features

No.	Feature	Description
1	start_time	Timestamp of connection start time.
2	src_ip	Source IP address of a connection.
3	src_zone	The topological zone of the source host {GLOBAL, MULTICAST, UNICAST, UNKNOWN, LOCAL, LAN1, LAN2, LAN3, LAN4, LAN5, LAN6}
4	srcprt	Source port number of a connection.
5	dst_ip	Destination IP address of a connection.
6	dst_zone	The topological zone of the destination host {GLOBAL, MULTICAST, UNICAST, UNKNOWN, LOCAL, LAN1, LAN2, LAN3, LAN4, LAN5, LAN6}
7	dstprt	Destination port number of a connection.
8	duration	Duration in seconds of a connection. This is the time difference between timestamps of the first and last packet of a connection.
9	ipVersion	IP version (IPv4 or IPv6) of a connection.
10	protocol	Transport protocol used for a connection (ICMP, TCP, UDP, etc)
11	conn_state	Bro label of connection state (13 different states in total). This will be the last known state of a connection ²² .
12	service	The application protocol of a connection as detected by Bro.
13	synthetic	One (1) if synthetic connection and zero (0) otherwise.

²² <https://www.bro.org/sphinx/scripts/base/protocols/conn/main.bro.html#type-Conn::Info>

No.	Feature	Description
14	origOrder	The original sequence number of connections and 0 for all synthetic connections
15	src_ip_bytes	Total IP bytes sent by source host.
16	dst_ip_bytes	Total IP bytes sent by destination host.
17	src_bytes	Total payload bytes sent by source host.
18	dst_bytes	Total payload bytes sent by destination host.
19	src_pkts	Total packets sent by source host.
20	dst_pkts	Total packets sent by destination host.
20a	wrong_fragment	<i>Number of wrong fragment packets as defined by Bro. There are 8 fragment related cases as identified by Bro internal implementation (excessively_large_fragment, excessively_small_fragment, fragment_inconsistency, fragment_overlap, fragment_size_inconsistency, fragment_protocol_inconsistency, incompletely_captured_fragment and fragment_with_DF)</i> ²³
21	urg	Number of total urgent TCP packets within a connection.
22	bro_duration	Duration in seconds of a connection as returned by Bro.
23	bro_conn_state	Bro label of connection state (13 different states in total). This will be the last known state of a connection. ²⁴
24	bro_service	The application protocol of a connection as detected by Bro's internal engine.
25	bro_src_ip_bytes	Total IP bytes sent by source host as detected by Bro's internal engine.
26	bro_dst_ip_bytes	Total IP bytes sent by destination host as detected by Bro's internal engine.
27	bro_src_bytes	Total payload bytes sent by source host as detected by Bro's internal engine.
28	bro_dst_bytes	Total payload bytes sent by destination host as detected by Bro's internal engine.
29	bro_src_pkts	Total packets sent by source host as detected by Bro's internal engine.
30	bro_dst_pkts	Total packets sent by destination host as detected by Bro's internal engine.
31	conn_start	Connection NORMAL start {0, 1}
32	conn_partial_start	Connection PARTIAL start {0, 1}. This is usually raised by Bro when it detects the start of a new active TCP connection without seeing the initial handshake ²⁵ .
33	conn_close	Connection NORMAL close {0, 1}
34	conn_partial_close	Connection PARTIAL close {0, 1}. This is usually raised by Bro when one of the communicating hosts attempt to close an inactive TCP connection with a FIN handshake or an RST packets ²⁵ .
35	conn_weird	Number of WEIRD events raised by a connection. This event is raised by Bro when an abnormal activity is detected within a certain connection ²⁶ .
36	conn_content_weird	Number of WEIRD events raised by the content of a connection.
37	conn_stats_orig_num_pkts	Total packets sent by source host as computed by Bro's statistics event ²⁷ .

²³ <https://github.com/bro/bro/blob/master/src/Frag.cc>

²⁴ <https://www.bro.org/sphinx/scripts/base/protocols/conn/main.bro.html#type-Conn::Info>

²⁵ https://www.bro.org/sphinx/scripts/base/bif/plugins/Bro_TCP.events.bif.bro.html

²⁶ <https://www.bro.org/sphinx/scripts/base/bif/event.bif.bro.html>

²⁷ https://www.bro.org/sphinx/scripts/base/init-bare.bro.html#type-endpoint_stats

No.	Feature	Description
38	conn_stats_orig_num_rxmit	Total retransmitted packets sent by source host as computed by Bro's statistics event.
39	conn_stats_orig_num_rxmit_bytes	Total retransmitted bytes sent by source host as computed by Bro's statistics event.
40	conn_stats_orig_num_in_order	Total in-order packets sent by source host as computed by Bro's statistics event.
41	conn_stats_orig_num_out_order	Total out-of-order packets sent by source host as computed by Bro's statistics event.
42	conn_stats_orig_num_repl	Total replicated packets (last packet was sent again) sent by source host as computed by Bro's statistics event.
43	conn_stats_orig_endian_type	Endian type used by the source host. { ENDIAN_UNKNOWN, ENDIAN_BIG, ENDIAN_LITTLE, ENDIAN_CONFUSED }
44	conn_stats_resp_num_pkts	Total packets sent by destination host as computed by Bro's statistics event ²⁸ .
45	conn_stats_resp_num_rxmit	Total retransmitted packets sent by destination host as computed by Bro's statistics event.
46	conn_stats_resp_num_rxmit_bytes	Total retransmitted bytes sent by destination host as computed by Bro's statistics event.
47	conn_stats_resp_num_in_order	Total in-order packets sent by destination host as computed by Bro's statistics event.
48	conn_stats_resp_num_out_order	Total out-of-order packets sent by destination host as computed by Bro's statistics event.
49	conn_stats_resp_num_repl	Total replicated packets (last packet was sent again) sent by destination host as computed by Bro's statistics event.
50	conn_stats_resp_endian_type	Endian type used by the destination host. { ENDIAN_UNKNOWN, ENDIAN_BIG, ENDIAN_LITTLE, ENDIAN_CONFUSED }
50a	<i>ip4_src_hl_change</i>	<i>Number of changes in Header Length field in source IPv4 packets within a connection ²⁹.</i>
51	ip4_src_hl_current	Last seen value of Header Length field in source IPv4 packet of a connection.
52	ip4_src_hl_max	Maximum value detected of Header Length field in source IPv4 packet of a connection.
53	ip4_src_hl_min	Minimum value detected of Header Length field in source IPv4 packet of a connection.
53a	<i>ip4_src_tos_change</i>	<i>Number of changes in Type of service field in source IPv4 packets within a connection.</i>
54	ip4_src_tos_current	Last seen value of Type of service field in source IPv4 packet of a connection.
55	ip4_src_tos_max	Maximum value detected of Type of service field in source IPv4 packet of a connection.
56	ip4_src_tos_min	Minimum value detected of Type of service field in source IPv4 packet of a connection.
57	ip4_src_len_change	Number of changes in packet Length field in source IPv4 packets within a connection.
58	ip4_src_len_current	Last seen value in packet Length field in source IPv4 packet of a connection.
59	ip4_src_len_max	Maximum value detected in packet Length field in source IPv4 packet of a connection.
60	ip4_src_len_min	Minimum value detected in packet Length field in source IPv4 packet of a connection.
61	ip4_src_ttl_change	Number of changes in Time To Live field in source IPv4 packets within a connection.
62	ip4_src_ttl_current	Last seen value in Time To Live field in source IPv4 packet of a connection.

²⁸ https://www.bro.org/sphinx/scripts/base/init-bare.bro.html#type-endpoint_stats

²⁹ https://www.bro.org/sphinx/scripts/base/init-bare.bro.html#type-ip4_hdr

No.	Feature	Description
63	ip4_src_ttl_max	Maximum value detected in Time To Live field in source IPv4 packet of a connection.
64	ip4_src_ttl_min	Minimum value detected in Time To Live field in source IPv4 packet of a connection.
64a	ip4_dst_hl_change	<i>Number of changes in Header Length field in destination IPv4 packets within a connection.</i>
65	ip4_dst_hl_current	Last seen value in Header Length field in destination IPv4 packet of a connection.
66	ip4_dst_hl_max	Maximum value detected in Header Length field in destination IPv4 packet of a connection.
67	ip4_dst_hl_min	Minimum value detected in Header Length field in destination IPv4 packet of a connection.
68	ip4_dst_tos_change	Number of changes in Type of service field in destination IPv4 packets within a connection.
69	ip4_dst_tos_current	Last seen value in Type of service field in destination IPv4 packet of a connection.
70	ip4_dst_tos_max	Maximum value detected in Type of service field in destination IPv4 packet of a connection.
71	ip4_dst_tos_min	Minimum value detected in Type of service field in destination IPv4 packet of a connection.
72	ip4_dst_len_change	Number of changes in packet Length field in destination IPv4 packets within a connection.
73	ip4_dst_len_current	Last seen value in packet Length field in destination IPv4 packet of a connection.
74	ip4_dst_len_max	Maximum value detected in packet Length field in destination IPv4 packet of a connection.
75	ip4_dst_len_min	Minimum value detected in packet Length field in destination IPv4 packet of a connection.
76	ip4_dst_ttl_change	Number of changes in Time To Live field in destination IPv4 packets within a connection.
77	ip4_dst_ttl_current	Last seen value in Time To Live field in destination IPv4 packet of a connection.
78	ip4_dst_ttl_max	Maximum value detected in Time To Live field in destination IPv4 packet of a connection.
79	ip4_dst_ttl_min	Minimum value detected in Time To Live field in destination IPv4 packet of a connection.
79a	ip6_src_class_change	<i>Number of changes in Traffic class field in source IPv6 packets within a connection³⁰.</i>
79b	ip6_src_class_current	<i>Last seen value of Traffic class field in source IPv6 packet of a connection.</i>
79c	ip6_src_class_max	<i>Maximum value detected of Traffic class field in source IPv6 packet of a connection.</i>
79d	ip6_src_class_min	<i>Minimum value detected of Traffic class field in source IPv6 packet of a connection.</i>
79e	ip6_src_flow_change	<i>Number of changes in Flow label field in source IPv6 packets within a connection.</i>
79f	ip6_src_flow_current	<i>Last seen value of Flow label field in source IPv6 packet of a connection.</i>
79g	ip6_src_flow_max	<i>Maximum value detected of Flow label field in source IPv6 packet of a connection.</i>
79h	ip6_src_flow_min	<i>Minimum value detected of Flow label field in source IPv6 packet of a connection.</i>
79i	ip6_src_len_change	<i>Number of changes in Payload length field in source IPv6 packets within a connection.</i>

³⁰ https://www.bro.org/sphinx/scripts/base/init-bare.bro.html#type-ip6_hdr

No.	Feature	Description
80	<code>ip6_src_len_current</code>	Last seen value of Payload length field in source IPv6 packet of a connection.
81	<code>ip6_src_len_max</code>	Maximum value detected of Payload length field in source IPv6 packet of a connection.
82	<code>ip6_src_len_min</code>	Minimum value detected of Payload length field in source IPv6 packet of a connection.
82a	<i><code>ip6_src_hlim_change</code></i>	<i>Number of changes in Hop limit field in source IPv6 packets within a connection.</i>
83	<code>ip6_src_hlim_current</code>	Last seen value of Hop limit field in source IPv6 packet of a connection.
84	<code>ip6_src_hlim_max</code>	Maximum value detected of Hop limit field in source IPv6 packet of a connection.
85	<code>ip6_src_hlim_min</code>	Minimum value detected of Hop limit field in source IPv6 packet of a connection.
86	<code>ip6_src_exts</code>	Boolean value to identify if any of source IPv6 packet of a connection is an Extension header chain.
86a	<i><code>ip6_dst_class_change</code></i>	<i>Number of changes in Traffic class field in destination IPv6 packets within a connection³¹.</i>
86b	<i><code>ip6_dst_class_current</code></i>	<i>Last seen value of Traffic class field in destination IPv6 packet of a connection.</i>
86c	<i><code>ip6_dst_class_max</code></i>	<i>Maximum value detected of Traffic class field in destination IPv6 packet of a connection.</i>
86d	<i><code>ip6_dst_class_min</code></i>	<i>Minimum value detected of Traffic class field in destination IPv6 packet of a connection.</i>
86e	<i><code>ip6_dst_flow_change</code></i>	<i>Number of changes in Flow label field in destination IPv6 packets within a connection.</i>
86f	<i><code>ip6_dst_flow_current</code></i>	<i>Last seen value of Flow label field in destination IPv6 packet of a connection.</i>
86g	<i><code>ip6_dst_flow_max</code></i>	<i>Maximum value detected in Flow label field in destination IPv6 packet of a connection.</i>
86h	<i><code>ip6_dst_flow_min</code></i>	<i>Minimum value detected in Flow label field in destination IPv6 packet of a connection.</i>
86i	<i><code>ip6_dst_len_change</code></i>	<i>Number of changes in Payload length field in destination IPv6 packets within a connection.</i>
86j	<i><code>ip6_dst_len_current</code></i>	<i>Last seen value of Payload length field in destination IPv6 packet of a connection.</i>
86k	<i><code>ip6_dst_len_max</code></i>	<i>Maximum value detected in Payload length field in destination IPv6 packet of a connection.</i>
86l	<i><code>ip6_dst_len_min</code></i>	<i>Minimum value detected in Payload length field in destination IPv6 packet of a connection.</i>
86m	<i><code>ip6_dst_hlim_change</code></i>	<i>Number of changes in Hop limit field in destination IPv6 packets within a connection.</i>
86n	<i><code>ip6_dst_hlim_current</code></i>	<i>Last seen value of Hop limit field in destination IPv6 packet of a connection.</i>
86o	<i><code>ip6_dst_hlim_max</code></i>	<i>Maximum value detected in Hop limit field in destination IPv6 packet of a connection.</i>
86p	<i><code>ip6_dst_hlim_min</code></i>	<i>Minimum value detected in Hop limit field in destination IPv6 packet of a connection.</i>
86q	<i><code>ip6_dst_exts</code></i>	<i>Boolean value to identify if any of destination IPv6 packet of a connection is an Extension header chain.</i>
87	<code>icmp_src_icmp_type</code>	Type of source ICMP packet.
87a	<i><code>icmp_dst_icmp_type</code></i>	<i>Type of destination ICMP packet.</i>
88	<code>tcp_src_hl_change</code>	Number of changes in header length field in source TCP packets within a connection.

³¹ https://www.bro.org/sphinx/scripts/base/init-bare.bro.html#type-ip6_hdr

No.	Feature	Description
89	tcp_src_hl_current	Last seen value in header length field in source TCP packet of a connection.
90	tcp_src_hl_max	Maximum value detected in header length field in source TCP packet of a connection.
91	tcp_src_hl_min	Minimum value detected in header length field in source TCP packet of a connection.
92	tcp_src_dl_change	Number of changes in data length field in source TCP packets within a connection.
93	tcp_src_dl_current	Last seen value in data length field in source TCP packet of a connection.
94	tcp_src_dl_max	Maximum value detected in data length field in source TCP packet of a connection.
95	tcp_src_dl_min	Minimum value detected in data length field in source TCP packet of a connection.
96	tcp_src_win_change	Number of changes in window field in source TCP packets within a connection.
97	tcp_src_win_current	Last seen value of window field in source TCP packet of a connection.
98	tcp_src_win_max	Maximum value detected in window field in source TCP packet of a connection.
99	tcp_src_win_min	Minimum value detected in window field in source TCP packet of a connection.
99a	tcp_src_flags_NS_flags	Total number of source TCP packets in a connection with NS flag.
99b	tcp_src_flags_CWR_flags	Total number of source TCP packets in a connection with Congestion Window Reduced (CWR) flag.
99c	tcp_src_flags_ECE_flags	Total number of source TCP packets in a connection with ECE flag.
99d	tcp_src_flags_URG_flags	Total number of source TCP packets in a connection with Urgent (URG) flag.
100	tcp_src_flags_ACK_flags	Total number of source TCP packets in a connection with Acknowledgment (ACK) flag.
101	tcp_src_flags_PSH_flags	Total number of source TCP packets in a connection with Push (PSH) flag.
102	tcp_src_flags_RST_flags	Total number of source TCP packets in a connection with Reset (RST) flag.
103	tcp_src_flags_SYN_flags	Total number of source TCP packets in a connection with Synchronize (SYN) flag.
104	tcp_src_flags_FIN_flags	Total number of source TCP packets in a connection with FIN flag.
104a	tcp_src_0_flags	Total number of source TCP packets in a connection with no flag is set.
105	tcp_src_1_flags	Total number of source TCP packets in a connection with 1 flag set.
106	tcp_src_2_flags	Total number of source TCP packets in a connection with 2 flags set.
107	tcp_src_3_flags	Total number of source TCP packets in a connection with 3 flags set.
107a	tcp_src_4_flags	Total number of source TCP packets in a connection with 4 flags set.
107b	tcp_src_5_flags	Total number of source TCP packets in a connection with 5 flags set.
107c	tcp_src_6_flags	Total number of source TCP packets in a connection with 6 flags set.
107d	tcp_src_7_flags	Total number of source TCP packets in a connection with 7 flags set.
107e	tcp_src_8_flags	Total number of source TCP packets in a connection with 8 flags set.

No.	Feature	Description
107f	<i>tcp_src_9_flags</i>	<i>Total number of source TCP packets in a connection with 9 (ALL) flags set.</i>
108	tcp_dst_hl_change	Number of changes in header length field in destination TCP packets within a connection.
109	tcp_dst_hl_current	Last seen value in header length field in destination TCP packet of a connection.
110	tcp_dst_hl_max	Maximum value detected in header length field in destination TCP packet of a connection.
111	tcp_dst_hl_min	Minimum value detected in header length field in destination TCP packet of a connection.
112	tcp_dst_dl_change	Number of changes in data length field in destination TCP packets within a connection.
113	tcp_dst_dl_current	Last seen value in data length field in destination TCP packet of a connection.
114	tcp_dst_dl_max	Maximum value detected in data length field in destination TCP packet of a connection.
115	tcp_dst_dl_min	Minimum value detected in data length field in destination TCP packet of a connection.
116	tcp_dst_win_change	Number of changes in window field in destination TCP packets within a connection.
117	tcp_dst_win_current	Last seen value in window field in destination TCP packet of a connection.
118	tcp_dst_win_max	Maximum value detected in window field in destination TCP packet of a connection.
119	tcp_dst_win_min	Minimum value detected in window field in destination TCP packet of a connection.
119a	<i>tcp_dst_flags_NS_flags</i>	<i>Total number of destination TCP packets in a connection with NS flag.</i>
120	tcp_dst_flags_CWR_flags	Total number of destination TCP packets in a connection with Congestion Window Reduced (CWR) flag.
120a	<i>tcp_dst_flags_ECE_flags</i>	<i>Total number of destination TCP packets in a connection with ECE flag.</i>
120b	<i>tcp_dst_flags_URG_flags</i>	<i>Total number of destination TCP packets in a connection with Urgent (URG) flag.</i>
121	tcp_dst_flags_ACK_flags	Total number of destination TCP packets in a connection with Acknowledgment (ACK) flag.
122	tcp_dst_flags_PSH_flags	Total number of destination TCP packets in a connection with Push (PSH) flag.
123	tcp_dst_flags_RST_flags	Total number of destination TCP packets in a connection with Reset (RST) flag.
124	tcp_dst_flags_SYN_flags	Total number of destination TCP packets in a connection with Synchronize (SYN) flag.
125	tcp_dst_flags_FIN_flags	Total number of destination TCP packets in a connection with FIN flag.
125a	<i>tcp_dst_0_flags</i>	<i>Total number of destination TCP packets in a connection with no flag is set.</i>
126	tcp_dst_1_flags	Total number of destination TCP packets in a connection with 1 flag set.
127	tcp_dst_2_flags	Total number of destination TCP packets in a connection with 2 flags set.
128	tcp_dst_3_flags	Total number of destination TCP packets in a connection with 3 flags set.
128a	<i>tcp_dst_4_flags</i>	<i>Total number of destination TCP packets in a connection with 4 flags set.</i>
128b	<i>tcp_dst_5_flags</i>	<i>Total number of destination TCP packets in a connection with 5 flags set.</i>
128c	<i>tcp_dst_6_flags</i>	<i>Total number of destination TCP packets in a connection with 6 flags set.</i>

No.	Feature	Description
128d	<i>tcp_dst_7_flags</i>	<i>Total number of destination TCP packets in a connection with 7 flags set.</i>
128e	<i>tcp_dst_8_flags</i>	<i>Total number of destination TCP packets in a connection with 8 flags set.</i>
128f	<i>tcp_dst_9_flags</i>	<i>Total number of destination TCP packets in a connection with 9 (ALL) flags set.</i>
129	udp_src_ulen_change	Number of changes in length field in source UDP packets within a connection.
130	udp_src_ulen_current	Last seen value in length field in source UDP packet of a connection.
131	udp_src_ulen_max	Maximum value detected in length field in source UDP packet of a connection.
132	udp_src_ulen_min	Minimum value detected in length field in source UDP packet of a connection.
133	udp_dst_ulen_change	Number of changes in length field in destination UDP packets within a connection.
134	udp_dst_ulen_current	Last seen value in length field in destination UDP packet of a connection.
135	udp_dst_ulen_max	Maximum value detected in length field in destination UDP packet of a connection.
136	udp_dst_ulen_min	Minimum value detected in length field in destination UDP packet of a connection.
137	conn_max_pkts_gap_time	Maximum time gap between exchanged packets within a connection.
138	conn_min_pkts_gap_time	Minimum time gap between exchanged packets within a connection.
139	src_max_pkts_gap_time	Maximum time gap between source packets within a connection.
140	src_min_pkts_gap_time	Minimum time gap between source packets within a connection.
141	src_total_pkts_gap_time	Total time gap between source packets within a connection.
142	dst_max_pkts_gap_time	Maximum time gap between destination packets within a connection.
143	dst_min_pkts_gap_time	Minimum time gap between destination packets within a connection.
144	dst_total_pkts_gap_time	Total time gap between destination packets within a connection.

D.2. Connection-Based Features

No.	Feature	Description
145	DFMC_totalConnections	Total number of connections in a 100 connection window.
146	DFMC_same_src_hosts_count	Number of connections with the same source host as a current connection in a 100 connection window.
147	DFMC_diff_src_hosts_count	Number of connections with a different source host to the current connection in a 100 connection window.
148	DFMC_same_dst_hosts_count	Number of connections with the same destination host as a current connection in a 100 connection window.
149	DFMC_diff_dst_hosts_count	Number of connections with a different destination host to the current connection in a 100 connection window.

No.	Feature	Description
150	DFMC_same_srv_count	Number of connections with the same service as a current connection in a 100 connection window.
151	DFMC_diff_srv_count	Number of connections with a different service to the current connection in a 100 connection window.
152	DFMC_same_src_hosts_PCT	Percentage of connections with the same source host as the current connection in a 100 connection window.
153	DFMC_diff_src_hosts_PCT	Percentage of connections with a different source host to the current connection in a 100 connection window.
154	DFMC_same_dst_hosts_PCT	Percentage of connections with the same destination host as a current connection in a 100 connection window.
155	DFMC_diff_dst_hosts_PCT	Percentage of connections with a different destination host to the current connection in a 100 connection window.
156	DFMC_same_srv_PCT	Percentage of connections with the same service as a current connection in a 100 connection window.
157	DFMC_diff_srv_PCT	Percentage of connections with a different service to the current connection in a 100 connection window.
158	DFMCB_1	Number of TCP connections between the same hosts as a current connection in a 100 connection window.
159	DFMCB_2	Number of UDP connections between the same hosts as a current connection in a 100 connection window.
160	DFMCB_3	Number of ICMP connections between the same hosts as a current connection in a 100 connection window.
161	DFMCB_4	Number of TCP connections with source IP=current source IP, destination IP=current destination IP and source port=current source port as the current connection in a 100 connection window.
162	DFMCB_5	Number of TCP connections with source IP=current source IP, destination IP=current destination IP and source port!=current source port as the current connection in a 100 connection window.
163	DFMCB_6	Number of TCP connections with source IP=current source IP, destination IP=current destination IP and source port=current destination port as the current connection in a 100 connection window.
164	DFMCB_7	Number of TCP connections with source IP=current source IP, destination IP=current destination IP and source port!=current destination port as the current connection in a 100 connection window.
165	DFMCB_8	Number of TCP connections with source IP=current destination IP, destination IP=current source IP and source port=current source port as the current connection in a 100 connection window.
166	DFMCB_9	Number of TCP connections with source IP=current destination IP, destination IP=current source IP and source port!=current source port as the current connection in a 100 connection window.
167	DFMCB_10	Number of TCP connections with source IP=current destination IP, destination IP=current source IP and source port=current destination port as the current connection in a 100 connection window.
168	DFMCB_11	Number of TCP connections with source IP=current destination IP, destination IP=current source IP and source port!=current destination port as the current connection in a 100 connection window.
169	DFMCB_12	Number of UDP connections with source IP=current source IP, destination IP=current destination IP and source port=current source port as the current connection in a 100 connection window.

No.	Feature	Description
170	DFMCB_13	Number of UDP connections with source IP=current source IP, destination IP=current destination IP and source port!=current source port as the current connection in a 100 connection window.
171	DFMCB_14	Number of UDP connections with source IP=current source IP, destination IP=current destination IP and source port=current destination port as the current connection in a 100 connection window.
172	DFMCB_15	Number of UDP connections with source IP=current source IP, destination IP=current destination IP and source port!=current destination port as the current connection in a 100 connection window.
173	DFMCB_16	Number of UDP connections with source IP=current destination IP, destination IP=current source IP and source port=current source port as the current connection in a 100 connection window.
174	DFMCB_17	Number of UDP connections with source IP=current destination IP, destination IP=current source IP and source port!=current source port as the current connection in a 100 connection window.
175	DFMCB_18	Number of UDP connections with source IP=current destination IP, destination IP=current source IP and source port=current destination port as the current connection in a 100 connection window.
176	DFMCB_19	Number of UDP connections with source IP=current destination IP, destination IP=current source IP and source port!=current destination port as the current connection in a 100 connection window.
177	DFMCB_20	Number of TCP packets with source IP=current destination IP and destination IP=current source IP as the current connection in a 100 connection window.
178	DFMCB_21	Number of TCP packets with source IP=current source IP and destination IP=current destination IP as the current connection in a 100 connection window.
179	DFMCB_22	Total TCP bytes of packets with source IP=current destination IP and destination IP=current source IP as the current connection in a 100 connection window.
180	DFMCB_23	Total TCP bytes of packets with source IP=current source IP and destination IP=current destination IP as the current connection in a 100 connection window.
181	DFMCB_23a	Average (DFMCB_22/DFMCB_20) TCP bytes of packets with source IP=current destination IP and destination IP=current source IP as the current connection in a 100 connection window.
182	DFMCB_23b	Average (DFMCB_23/DFMCB_21) TCP bytes of packets with source IP=current source IP and destination IP=current destination IP as the current connection in a 100 connection window.
183	DFMCB_24	Number of UDP packets with source IP=current destination IP and destination IP=current source IP as the current connection in a 100 connection window.
184	DFMCB_25	Number of UDP packets with source IP=current source IP and destination IP=current destination IP as the current connection in a 100 connection window.
185	DFMCB_26	Total UDP bytes of packets with source IP=current destination IP and destination IP=current source IP as the current connection in a 100 connection window.

No.	Feature	Description
186	DFMCB_27	Total UDP bytes of packets with source IP=current source IP and destination IP=current destination IP as the current connection in a 100 connection window.
187	DFMCB_27a	Average (DFMCB_26/DFMCB_24) UDP bytes of packets with source IP=current destination IP and destination IP=current source IP as the current connection in a 100 connection window.
188	DFMCB_27b	Average (DFMCB_27/DFMCB_25) UDP bytes of packets with source IP=current source IP and destination IP=current destination IP as the current connection in a 100 connection window.
189	DFMCB_28	Number of ICMP packets with source IP=current destination IP and destination IP=current source IP as the current connection in a 100 connection window.
190	DFMCB_29	Number of ICMP packets with source IP=current source IP and destination IP=current destination IP as the current connection in a 100 connection window.
191	DFMCB_30	Total ICMP bytes of packets with source IP=current destination IP and destination IP=current source IP as the current connection in a 100 connection window.
192	DFMCB_31	Total ICMP bytes of packets with source IP=current source IP and destination IP=current destination IP as the current connection in a 100 connection window.
193	DFMCB_31a	Average (DFMCB_30/DFMCB_28) ICMP bytes of packets with source IP=current destination IP and destination IP=current source IP as the current connection in a 100 connection window.
194	DFMCB_31b	Average (DFMCB_31/DFMCB_29) ICMP bytes of packets with source IP=current source IP and destination IP=current destination IP as the current connection in a 100 connection window.
194a	DFMCB_32	<i>Number of TCP (URG) packets with source IP=current source IP and destination IP=current destination IP as the current connection in a 100 connection window.</i>
195	DFMCB_33	Number of TCP (ACK) packets with source IP=current source IP and destination IP=current destination IP as the current connection in a 100 connection window.
196	DFMCB_34	Number of TCP (PSH) packets with source IP=current source IP and destination IP=current destination IP as the current connection in a 100 connection window.
197	DFMCB_35	Number of TCP (RST) packets with source IP=current source IP and destination IP=current destination IP as the current connection in a 100 connection window.
198	DFMCB_36	Number of TCP (SYN) packets with source IP=current source IP and destination IP=current destination IP as the current connection in a 100 connection window.
199	DFMCB_37	Number of TCP (FIN) packets with source IP=current source IP and destination IP=current destination IP as the current connection in a 100 connection window.
199a	DFMCB_38	<i>Number of TCP (URG) packets with source IP=current destination IP and destination IP=current source IP as the current connection in a 100 connection window.</i>
200	DFMCB_39	Number of TCP (ACK) packets with source IP=current destination IP and destination IP=current source IP as the current connection in a 100 connection window.
201	DFMCB_40	Number of TCP (PSH) packets with source IP=current destination IP and destination IP=current source IP as the current connection in a 100 connection window.

No.	Feature	Description
202	DFMCB_41	Number of TCP (RST) packets with source IP=current destination IP and destination IP=current source IP as the current connection in a 100 connection window.
203	DFMCB_42	Number of TCP (SYN) packets with source IP=current destination IP and destination IP=current source IP as the current connection in a 100 connection window.
204	DFMCB_43	Number of TCP (FIN) packets with source IP=current destination IP and destination IP=current source IP as the current connection in a 100 connection window.
204a	DFMCB_44	<i>Number of Echo (type 8) ICMP packets with source IP=current source IP and destination IP=current destination IP as the current connection in a 100 connection window.</i>
204b	DFMCB_45	<i>Number of Echo (type 8) ICMP packets with source IP=current destination IP and destination IP=current source IP as the current connection in a 100 connection window.</i>
204c	DFMCB_46	<i>Number of Destination Unreachable (type 3) ICMP packets with source IP=current source IP and destination IP=current destination IP as the current connection in a 100 connection window.</i>
204d	DFMCB_47	<i>Number of Destination Unreachable (type 3) ICMP packets with source IP=current destination IP and destination IP=current source IP as the current connection in a 100 connection window.</i>
205	DFMCOS_1	Number of TCP connections with source IP=current source IP in a 100 connection window.
206	DFMCOS_2	Number of UDP connections with source IP=current source IP in a 100 connection window.
207	DFMCOS_3	Number of ICMP connections with source IP=current source IP in a 100 connection window.
208	DFMCOS_4	Number of TCP connections with destination IP=current source IP in a 100 connection window.
209	DFMCOS_5	Number of UDP connections with destination IP=current source IP in a 100 connection window.
210	DFMCOS_6	Number of ICMP connections with destination IP=current source IP in a 100 connection window.
211	DFMCOS_7	Number of TCP connections with source IP=current source IP and destination port=current destination port in a 100 connection window.
212	DFMCOS_8	Number of TCP connections with source IP=current source IP and destination port!=current destination port in a 100 connection window.
213	DFMCOS_9	Number of TCP connections with destination IP=current source IP and destination port=current destination port in a 100 connection window.
214	DFMCOS_10	Number of TCP connections with destination IP=current source IP and destination port!=current destination port in a 100 connection window.
215	DFMCOS_11	Number of TCP connections with source IP=current source IP and destination port=current source port in a 100 connection window.
216	DFMCOS_12	Number of TCP connections with source IP=current source IP and destination port!=current source port in a 100 connection window.
217	DFMCOS_13	Number of TCP connections with destination IP=current source IP and destination port=current source port in a 100 connection window.

No.	Feature	Description
218	DFMCOS_14	Number of TCP connections with destination IP=current source IP and destination port!=current source port in a 100 connection window.
219	DFMCOS_15	Number of UDP connections with source IP=current source IP and destination port=current destination port in a 100 connection window.
220	DFMCOS_16	Number of UDP connections with source IP=current source IP and destination port!=current destination port in a 100 connection window.
221	DFMCOS_17	Number of UDP connections with destination IP=current source IP and destination port=current destination port in a 100 connection window.
222	DFMCOS_18	Number of UDP connections with destination IP=current source IP and destination port!=current destination port in a 100 connection window.
223	DFMCOS_19	Number of UDP connections with source IP=current source IP and destination port=current source port in a 100 connection window.
224	DFMCOS_20	Number of UDP connections with source IP=current source IP and destination port!=current source port in a 100 connection window.
225	DFMCOS_21	Number of UDP connections with destination IP=current source IP and destination port=current source port in a 100 connection window.
226	DFMCOS_22	Number of UDP connections with destination IP=current source IP and destination port!=current source port in a 100 connection window.
227	DFMCOS_23	Number of TCP connections with SYN packets where source IP=current source IP in a 100 connection window.
228	DFMCOS_24	Number of TCP connections with SYN packets where destination IP=current source IP in a 100 connection window.
229	DFMCOS_25	Number of TCP connections with RST packets where source IP=current source IP in a 100 connection window.
230	DFMCOS_26	Number of TCP connections with RST packets where destination IP=current source IP in a 100 connection window.
231	DFMCOS_27	Number of TCP packets with destination IP=current source IP in a 100 connection window.
232	DFMCOS_28	Number of TCP packets with source IP=current source IP in a 100 connection window.
233	DFMCOS_29	Total bytes of TCP packets with destination IP=current source IP in a 100 connection window.
234	DFMCOS_30	Total bytes of TCP packets with source IP=current source IP in a 100 connection window.
235	DFMCOS_30a	Average (DFMCOS_29/DFMCOS_27) TCP bytes of packets with destination IP=current source IP as the current connection in a 100 connection window.
236	DFMCOS_30b	Average (DFMCOS_30/DFMCOS_28) TCP bytes of packets with source IP=current source IP as the current connection in a 100 connection window.
237	DFMCOS_31	Number of UDP packets with destination IP=current source IP in a 100 connection window.
238	DFMCOS_32	Number of UDP packets with source IP=current source IP in a 100 connection window.
239	DFMCOS_33	Total bytes of UDP packets with destination IP=current source IP in a 100 connection window.
240	DFMCOS_34	Total bytes of UDP packets with source IP=current source IP in a 100 connection window.

No.	Feature	Description
241	DFMCOS_34a	Average (DFMCOS_33/DFMCOS_31) UDP bytes of packets with destination IP=current source IP as the current connection in a 100 connection window.
242	DFMCOS_34b	Average (DFMCOS_34/DFMCOS_32) UDP bytes of packets with source IP=current source IP as the current connection in a 100 connection window.
243	DFMCOS_35	Number of ICMP packets with destination IP=current source IP in a 100 connection window.
244	DFMCOS_36	Number of ICMP packets with source IP=current source IP in a 100 connection window.
245	DFMCOS_37	Total bytes of ICMP packets with destination IP=current source IP in a 100 connection window.
246	DFMCOS_38	Total bytes of ICMP packets with source IP=current source IP in a 100 connection window.
247	DFMCOS_38a	Average (DFMCOS_37/DFMCOS_35) ICMP bytes of packets with destination IP=current source IP as the current connection in a 100 connection window.
248	DFMCOS_38b	Average (DFMCOS_38/DFMCOS_36) ICMP bytes of packets with source IP=current source IP as the current connection in a 100 connection window.
249	DFMCOS_39	Number of Destination Unreachable (Type 3) ICMP packets with destination IP=current source IP in a 100 connection window.
249a	<i>DFMCOS_40</i>	<i>Number of Echo Reply (Type 0) ICMP packets with source IP=current source IP in a 100 connection window.</i>
249b	<i>DFMCOS_41</i>	<i>Number of TCP (URG) packets with destination IP=current source IP as the current connection in a 100 connection window.</i>
250	DFMCOS_42	Number of TCP (ACK) packets with destination IP=current source IP as the current connection in a 100 connection window.
251	DFMCOS_43	Number of TCP (PSH) packets with destination IP=current source IP as the current connection in a 100 connection window.
252	DFMCOS_44	Number of TCP (RST) packets with destination IP=current source IP as the current connection in a 100 connection window.
253	DFMCOS_45	Number of TCP (SYN) packets with destination IP=current source IP as the current connection in a 100 connection window.
254	DFMCOS_46	Number of TCP (FIN) packets with destination IP=current source IP as the current connection in a 100 connection window.
254a	<i>DFMCOS_47</i>	<i>Number of TCP (URG) packets with source IP=current source IP as the current connection in a 100 connection window.</i>
255	DFMCOS_48	Number of TCP (ACK) packets with source IP=current source IP as the current connection in a 100 connection window.
256	DFMCOS_49	Number of TCP (PSH) packets with source IP=current source IP as the current connection in a 100 connection window.
257	DFMCOS_50	Number of TCP (RST) packets with source IP=current source IP as the current connection in a 100 connection window.
258	DFMCOS_51	Number of TCP (SYN) packets with source IP=current source IP as the current connection in a 100 connection window.
259	DFMCOS_52	Number of TCP (FIN) packets with source IP=current source IP as the current connection in a 100 connection window.
260	DFMCOD_1	Number of TCP connections with source IP=current destination IP in a 100 connection window.

No.	Feature	Description
261	DFM COD_2	Number of UDP connections with source IP=current destination IP in a 100 connection window.
262	DFM COD_3	Number of ICMP connections with source IP=current destination IP in a 100 connection window.
263	DFM COD_4	Number of TCP connections with destination IP=current destination IP in a 100 connection window.
264	DFM COD_5	Number of UDP connections with destination IP=current destination IP in a 100 connection window.
265	DFM COD_6	Number of ICMP connections with destination IP=current destination IP in a 100 connection window.
266	DFM COD_7	Number of TCP connections with source IP=current destination IP and destination port=current destination port in a 100 connection window.
267	DFM COD_8	Number of TCP connections with source IP=current destination IP and destination port!=current destination port in a 100 connection window.
268	DFM COD_9	Number of TCP connections with destination IP=current destination IP and destination port=current destination port in a 100 connection window.
269	DFM COD_10	Number of TCP connections with destination IP=current destination IP and destination port!=current destination port in a 100 connection window.
270	DFM COD_11	Number of TCP connections with source IP=current destination IP and destination port=current source port in a 100 connection window.
271	DFM COD_12	Number of TCP connections with source IP=current destination IP and destination port!=current source port in a 100 connection window.
272	DFM COD_13	Number of TCP connections with destination IP=current destination IP and destination port=current source port in a 100 connection window.
273	DFM COD_14	Number of TCP connections with destination IP=current destination IP and destination port!=current source port in a 100 connection window.
274	DFM COD_15	Number of UDP connections with source IP=current destination IP and destination port=current destination port in a 100 connection window.
275	DFM COD_16	Number of UDP connections with source IP=current destination IP and destination port!=current destination port in a 100 connection window.
276	DFM COD_17	Number of UDP connections with destination IP=current destination IP and destination port=current destination port in a 100 connection window.
277	DFM COD_18	Number of UDP connections with destination IP=current destination IP and destination port!=current destination port in a 100 connection window.
278	DFM COD_19	Number of UDP connections with source IP=current destination IP and destination port=current source port in a 100 connection window.
279	DFM COD_20	Number of UDP connections with source IP=current destination IP and destination port!=current source port in a 100 connection window.
280	DFM COD_21	Number of UDP connections with destination IP=current destination IP and destination port=current source port in a 100 connection window.
281	DFM COD_22	Number of UDP connections with destination IP=current destination IP and destination port!=current source port in a 100 connection window.

No.	Feature	Description
282	DFM COD_23	Number of TCP connections with SYN packets where source IP=current destination IP in a 100 connection window.
283	DFM COD_24	Number of TCP connections with SYN packets where destination IP=current destination IP in a 100 connection window.
284	DFM COD_25	Number of TCP connections with RST packets where source IP=current destination IP in a 100 connection window.
285	DFM COD_26	Number of TCP connections with RST packets where destination IP=current destination IP in a 100 connection window.
286	DFM COD_27	Number of TCP packets with destination IP=current destination IP in a 100 connection window.
287	DFM COD_28	Number of TCP packets with source IP=current destination IP in a 100 connection window.
288	DFM COD_29	Total bytes of TCP packets with destination IP=current destination IP in a 100 connection window.
289	DFM COD_30	Total bytes of TCP packets with source IP=current destination IP in a 100 connection window.
290	DFM COD_30a	Average (DFM COS_29/DFM COS_27) TCP bytes of packets with destination IP=current destination IP as the current connection in a 100 connection window.
291	DFM COD_30b	Average (DFM COS_30/DFM COS_28) TCP bytes of packets with source IP=current destination IP as the current connection in a 100 connection window.
292	DFM COD_31	Number of UDP packets with destination IP=current destination IP in a 100 connection window.
293	DFM COD_32	Number of UDP packets with source IP=current destination IP in a 100 connection window.
294	DFM COD_33	Total bytes of UDP packets with destination IP=current destination IP in a 100 connection window.
295	DFM COD_34	Total bytes of UDP packets with source IP=current destination IP in a 100 connection window.
296	DFM COD_34a	Average (DFM COS_33/DFM COS_31) UDP bytes of packets with destination IP=current destination IP as the current connection in a 100 connection window.
297	DFM COD_34b	Average (DFM COS_34/DFM COS_32) UDP bytes of packets with source IP=current destination IP as the current connection in a 100 connection window.
298	DFM COD_35	Number of ICMP packets with destination IP=current destination IP in a 100 connection window.
299	DFM COD_36	Number of ICMP packets with source IP=current destination IP in a 100 connection window.
300	DFM COD_37	Total bytes of ICMP packets with destination IP=current destination IP in a 100 connection window.
301	DFM COD_38	Total bytes of ICMP packets with source IP=current destination IP in a 100 connection window.
302	DFM COD_38a	Average (DFM COS_37/DFM COS_35) ICMP bytes of packets with destination IP=current destination IP as the current connection in a 100 connection window.
303	DFM COD_38b	Average (DFM COS_38/DFM COS_36) ICMP bytes of packets with source IP=current destination IP as the current connection in a 100 connection window.
304	DFM COD_39	Number of Destination Unreachable (Type 3) ICMP packets with destination IP=current destination IP in a 100 connection window.
304a	DFM COD_40	<i>Number of Echo Reply (Type 0) ICMP packets with source IP=current destination IP in a 100 connection window.</i>

No.	Feature	Description
304b	DFMCO _D _41	<i>Number of TCP (URG) packets with destination IP=current destination IP as the current connection in a 100 connection window.</i>
305	DFMCO _D _42	Number of TCP (ACK) packets with destination IP=current destination IP as the current connection in a 100 connection window.
306	DFMCO _D _43	Number of TCP (PSH) packets with destination IP=current destination IP as the current connection in a 100 connection window.
307	DFMCO _D _44	Number of TCP (RST) packets with destination IP=current destination IP as the current connection in a 100 connection window.
308	DFMCO _D _45	Number of TCP (SYN) packets with destination IP=current destination IP as the current connection in a 100 connection window.
309	DFMCO _D _46	Number of TCP (FIN) packets with destination IP=current destination IP as the current connection in a 100 connection window.
309a	DFMCO _D _47	<i>Number of TCP (URG) packets with source IP=current destination IP as the current connection in a 100 connection window.</i>
310	DFMCO _D _48	Number of TCP (ACK) packets with source IP=current destination IP as the current connection in a 100 connection window.
311	DFMCO _D _49	Number of TCP (PSH) packets with source IP=current destination IP as the current connection in a 100 connection window.
312	DFMCO _D _50	Number of TCP (RST) packets with source IP=current destination IP as the current connection in a 100 connection window.
313	DFMCO _D _51	Number of TCP (SYN) packets with source IP=current destination IP as the current connection in a 100 connection window.
314	DFMCO _D _52	Number of TCP (FIN) packets with source IP=current destination IP as the current connection in a 100 connection window.
315	DFMCG_1	Total number of TCP connections in a 100 connection window.
316	DFMCG_2	Total number of UDP connections in a 100 connection window.
317	DFMCG_3	Total number of ICMP connections in a 100 connection window.
318	DFMCG_4	Number of TCP connections with source or destination port=current destination port in a 100 connection window.
319	DFMCG_5	Number of TCP connections with source or destination port!=current destination port in a 100 connection window.
320	DFMCG_6	Number of UDP connections with source or destination port=current destination port in a 100 connection window.
321	DFMCG_7	Number of UDP connections with source or destination port!=current destination port in a 100 connection window.
322	DFMCG_8	Total number of exchanged TCP packets in a 100 connection window.
323	DFMCG_9	Total number of exchanged UDP packets in a 100 connection window.
324	DFMCG_10	Total number of exchanged ICMP packets in a 100 connection window.
325	DFMCG_11	Number of TCP connections with destination port=current source port in a 100 connection window.

No.	Feature	Description
326	DFMCG_12	Number of UDP connections with destination port=current source port in a 100 connection window.
327	DFMCG_13	Number of TCP connections with source port=current source port in a 100 connection window.
328	DFMCG_14	Number of UDP connections with source port=current source port in a 100 connection window.
329	DFMCG_15	Number of TCP connections with destination port=current destination port in a 100 connection window.
330	DFMCG_16	Number of UDP connections with destination port=current destination port in a 100 connection window.
331	DFMCG_17	Number of TCP connections with source port=current destination port in a 100 connection window.
332	DFMCG_18	Number of UDP connections with source port=current destination port in a 100 connection window.
332a	DFMCG_19	<i>Number of TCP (URG) packets in a 100 connection window.</i>
333	DFMCG_20	Number of TCP (ACK) packets in a 100 connection window.
334	DFMCG_21	Number of TCP (PSH) packets in a 100 connection window.
335	DFMCG_22	Number of TCP (RST) packets in a 100 connection window.
336	DFMCG_23	Number of TCP (SYN) packets in a 100 connection window.
337	DFMCG_24	Number of TCP (FIN) packets in a 100 connection window.
338	DFMCG_25	Number of Destination Unreachable (Type 3) ICMP packets in a 100 connection window.
338a	DFMCG_26	<i>Number of Echo Reply (Type 0) ICMP packets in a 100 connection window.</i>
338b	DFMCG_27	<i>Number of Destination Unreachable (Type 3) ICMP connections in a 100 connection window.</i>
338c	DFMCG_28	<i>Number of Echo Reply (Type 0) ICMP connections in a 100 connection window.</i>
339	DFMCG_29	Number of TCP connection with SYN packets and destination port=current source port in a 100 connection window.
340	DFMCG_30	Number of TCP connection with SYN packets and source port=current source port in a 100 connection window.
341	DFMCG_31	Number of TCP connection with SYN packets and destination port=current destination port in a 100 connection window.
342	DFMCG_32	Number of TCP connection with SYN packets and source port=current destination port in a 100 connection window.
343	DFMCG_33	Number of TCP connection with RST packets and destination port=current source port in a 100 connection window.
344	DFMCG_34	Number of TCP connection with RST packets and source port=current source port in a 100 connection window.
345	DFMCG_35	Number of TCP connection with RST packets and destination port=current destination port in a 100 connection window.
346	DFMCG_36	Number of TCP connection with RST packets and source port=current destination port in a 100 connection window.
347	DFMCG_37	Total number of TCP connections with SYN packets in a 100 connection window.
348	DFMCG_38	Total number of TCP connections with RST packets in a 100 connection window.

D.3. Time-Based Features

No.	Feature	Description
349	DFMT_totalConnections	Total number of connections in a 5 second window.
350	DFMT_same_src_hosts_count	Number of connections with the same source host as a current connection in a 5 second window.
351	DFMT_diff_src_hosts_count	Number of connections with a different source host to the current connection in a 5 second window.
352	DFMT_same_dst_hosts_count	Number of connections with the same destination host as a current connection in a 5 second window.
353	DFMT_diff_dst_hosts_count	Number of connections with a different destination host to the current connection in a 5 second window.
354	DFMT_same_srv_count	Number of connections with the same service as a current connection in a 5 second window.
355	DFMT_diff_srv_count	Number of connections with a different service to the current connection in a 5 second window.
356	DFMT_same_src_hosts_PCT	Percentage of connections with the same source host as a current connection in a 5 second window.
357	DFMT_diff_src_hosts_PCT	Percentage of connections with a different source host to the current connection in a 5 second window.
358	DFMT_same_dst_hosts_PCT	Percentage of connections with the same destination host as a current connection in a 5 second window.
359	DFMT_diff_dst_hosts_PCT	Percentage of connections with a different destination host to the current connection in a 5 second window.
360	DFMT_same_srv_PCT	Percentage of connections with the same service as a current connection in a 5 second window.
361	DFMT_diff_srv_PCT	Percentage of connections with a different service to the current connection in a 5 second window.
362	DFMTB_1	Number of TCP connections between the same hosts as a current connection in a 5 second window.
363	DFMTB_2	Number of UDP connections between the same hosts as a current connection in a 5 second window.
364	DFMTB_3	Number of ICMP connections between the same hosts as a current connection in a 5 second window.
365	DFMTB_4	Number of TCP connections with source IP=current source IP, destination IP=current destination IP and source port=current source port as the current connection in a 5 second window.
366	DFMTB_5	Number of TCP connections with source IP=current source IP, destination IP=current destination IP and source port!=current source port as the current connection in a 5 second window.
367	DFMTB_6	Number of TCP connections with source IP=current source IP, destination IP=current destination IP and source port=current destination port as the current connection in a 5 second window.
368	DFMTB_7	Number of TCP connections with source IP=current source IP, destination IP=current destination IP and source port!=current destination port as the current connection in a 5 second window.
369	DFMTB_8	Number of TCP connections with source IP=current destination IP, destination IP=current source IP and source port=current source port as the current connection in a 5 second window.
370	DFMTB_9	Number of TCP connections with source IP=current destination IP, destination IP=current source IP and source

No.	Feature	Description
		port!=current source port as the current connection in a 5 second window.
371	DFMTB_10	Number of TCP connections with source IP=current destination IP, destination IP=current source IP and source port=current destination port as the current connection in a 5 second window.
372	DFMTB_11	Number of TCP connections with source IP=current destination IP, destination IP=current source IP and source port!=current destination port as the current connection in a 5 second window.
372a	DFMTB_12	<i>Number of UDP connections with source IP=current source IP, destination IP=current destination IP and source port=current source port as the current connection in a 5 second window.</i>
373	DFMTB_13	Number of UDP connections with source IP=current source IP, destination IP=current destination IP and source port!=current source port as the current connection in a 5 second window.
374	DFMTB_14	Number of UDP connections with source IP=current source IP, destination IP=current destination IP and source port=current destination port as the current connection in a 5 second window.
375	DFMTB_15	Number of UDP connections with source IP=current source IP, destination IP=current destination IP and source port!=current destination port as the current connection in a 5 second window.
375a	DFMTB_16	<i>Number of UDP connections with source IP=current destination IP, destination IP=current source IP and source port=current source port as the current connection in a 5 second window.</i>
376	DFMTB_17	Number of UDP connections with source IP=current destination IP, destination IP=current source IP and source port!=current source port as the current connection in a 5 second window.
376a	DFMTB_18	<i>Number of UDP connections with source IP=current destination IP, destination IP=current source IP and source port=current destination port as the current connection in a 5 second window.</i>
377	DFMTB_19	Number of UDP connections with source IP=current destination IP, destination IP=current source IP and source port!=current destination port as the current connection in a 5 second window.
378	DFMTB_20	Number of TCP packets with source IP=current destination IP and destination IP=current source IP as the current connection in a 5 second window.
379	DFMTB_21	Number of TCP packets with source IP=current source IP and destination IP=current destination IP as the current connection in a 5 second window.
380	DFMTB_22	Total TCP bytes of packets with source IP=current destination IP and destination IP=current source IP as the current connection in a 5 second window.
381	DFMTB_23	Total TCP bytes of packets with source IP=current source IP and destination IP=current destination IP as the current connection in a 5 second window.
382	DFMTB_23a	Average (DFMCB_22/DFMCB_20) TCP bytes of packets with source IP=current destination IP and destination IP=current source IP as the current connection in a 5 second window.

No.	Feature	Description
383	DFMTB_23b	Average (DFMFCB_23/DFMFCB_21) TCP bytes of packets with source IP=current source IP and destination IP=current destination IP as the current connection in a 5 second window.
384	DFMTB_24	Number of UDP packets with source IP=current destination IP and destination IP=current source IP as the current connection in a 5 second window.
385	DFMTB_25	Number of UDP packets with source IP=current source IP and destination IP=current destination IP as the current connection in a 5 second window.
386	DFMTB_26	Total UDP bytes of packets with source IP=current destination IP and destination IP=current source IP as the current connection in a 5 second window.
387	DFMTB_27	Total UDP bytes of packets with source IP=current source IP and destination IP=current destination IP as the current connection in a 5 second window.
388	DFMTB_27a	Average (DFMFCB_26/DFMFCB_24) UDP bytes of packets with source IP=current destination IP and destination IP=current source IP as the current connection in a 5 second window.
389	DFMTB_27b	Average (DFMFCB_27/DFMFCB_25) UDP bytes of packets with source IP=current source IP and destination IP=current destination IP as the current connection in a 5 second window.
390	DFMTB_28	Number of ICMP packets with source IP=current destination IP and destination IP=current source IP as the current connection in a 5 second window.
391	DFMTB_29	Number of ICMP packets with source IP=current source IP and destination IP=current destination IP as the current connection in a 5 second window.
392	DFMTB_30	Total ICMP bytes of packets with source IP=current destination IP and destination IP=current source IP as the current connection in a 5 second window.
393	DFMTB_31	Total ICMP bytes of packets with source IP=current source IP and destination IP=current destination IP as the current connection in a 5 second window.
394	DFMTB_31a	Average (DFMFCB_30/DFMFCB_28) ICMP bytes of packets with source IP=current destination IP and destination IP=current source IP as the current connection in a 5 second window.
395	DFMTB_31b	Average (DFMFCB_31/DFMFCB_29) ICMP bytes of packets with source IP=current source IP and destination IP=current destination IP as the current connection in a 5 second window.
395a	DFMTB_32	<i>Number of TCP (URG) packets with source IP=current source IP and destination IP=current destination IP as the current connection in a 5 second window.</i>
396	DFMTB_33	Number of TCP (ACK) packets with source IP=current source IP and destination IP=current destination IP as the current connection in a 5 second window.
397	DFMTB_34	Number of TCP (PSH) packets with source IP=current source IP and destination IP=current destination IP as the current connection in a 5 second window.
398	DFMTB_35	Number of TCP (RST) packets with source IP=current source IP and destination IP=current destination IP as the current connection in a 5 second window.
399	DFMTB_36	Number of TCP (SYN) packets with source IP=current source IP and destination IP=current destination IP as the current connection in a 5 second window.

No.	Feature	Description
400	DFMTB_37	Number of TCP (FIN) packets with source IP=current source IP and destination IP=current destination IP as the current connection in a 5 second window.
400a	<i>DFMTB_38</i>	<i>Number of TCP (URG) packets with source IP=current destination IP and destination IP=current source IP as the current connection in a 5 second window.</i>
401	DFMTB_39	Number of TCP (ACK) packets with source IP=current destination IP and destination IP=current source IP as the current connection in a 5 second window.
402	DFMTB_40	Number of TCP (PSH) packets with source IP=current destination IP and destination IP=current source IP as the current connection in a 5 second window.
403	DFMTB_41	Number of TCP (RST) packets with source IP=current destination IP and destination IP=current source IP as the current connection in a 5 second window.
404	DFMTB_42	Number of TCP (SYN) packets with source IP=current destination IP and destination IP=current source IP as the current connection in a 5 second window.
405	DFMTB_43	Number of TCP (FIN) packets with source IP=current destination IP and destination IP=current source IP as the current connection in a 5 second window.
405a	<i>DFMTB_44</i>	<i>Number of Echo (type 8) ICMP packets with source IP=current source IP and destination IP=current destination IP as the current connection in a 5 second window.</i>
405b	<i>DFMTB_45</i>	<i>Number of Echo (type 8) ICMP packets with source IP=current destination IP and destination IP=current source IP as the current connection in a 5 second window.</i>
405c	<i>DFMTB_46</i>	<i>Number of Destination Unreachable (type 3) ICMP packets with source IP=current source IP and destination IP=current destination IP as the current connection in a 5 second window.</i>
405d	<i>DFMTB_47</i>	<i>Number of Destination Unreachable (type 3) ICMP packets with source IP=current destination IP and destination IP=current source IP as the current connection in a 5 second window.</i>
406	DFMTOS_1	Number of TCP connections with source IP=current source IP in a 5 second window.
407	DFMTOS_2	Number of UDP connections with source IP=current source IP in a 5 second window.
408	DFMTOS_3	Number of ICMP connections with source IP=current source IP in a 5 second window.
409	DFMTOS_4	Number of TCP connections with destination IP=current source IP in a 5 second window.
410	DFMTOS_5	Number of UDP connections with destination IP=current source IP in a 5 second window.
411	DFMTOS_6	Number of ICMP connections with destination IP=current source IP in a 5 second window.
412	DFMTOS_7	Number of TCP connections with source IP=current source IP and destination port=current destination port in a 5 second window.
413	DFMTOS_8	Number of TCP connections with source IP=current source IP and destination port!=current destination port in a 5 second window.
414	DFMTOS_9	Number of TCP connections with destination IP=current source IP and destination port=current destination port in a 5 second window.
415	DFMTOS_10	Number of TCP connections with destination IP=current source IP and destination port!=current destination port in a 5 second window.

No.	Feature	Description
416	DFMTOS_11	Number of TCP connections with source IP=current source IP and destination port=current source port in a 5 second window.
417	DFMTOS_12	Number of TCP connections with source IP=current source IP and destination port!=current source port in a 5 second window.
418	DFMTOS_13	Number of TCP connections with destination IP=current source IP and destination port=current source port in a 5 second window.
419	DFMTOS_14	Number of TCP connections with destination IP=current source IP and destination port!=current source port in a 5 second window.
420	DFMTOS_15	Number of UDP connections with source IP=current source IP and destination port=current destination port in a 5 second window.
421	DFMTOS_16	Number of UDP connections with source IP=current source IP and destination port!=current destination port in a 5 second window.
422	DFMTOS_17	Number of UDP connections with destination IP=current source IP and destination port=current destination port in a 5 second window.
423	DFMTOS_18	Number of UDP connections with destination IP=current source IP and destination port!=current destination port in a 5 second window.
424	DFMTOS_19	Number of UDP connections with source IP=current source IP and destination port=current source port in a 5 second window.
425	DFMTOS_20	Number of UDP connections with source IP=current source IP and destination port!=current source port in a 5 second window.
426	DFMTOS_21	Number of UDP connections with destination IP=current source IP and destination port=current source port in a 5 second window.
427	DFMTOS_22	Number of UDP connections with destination IP=current source IP and destination port!=current source port in a 5 second window.
428	DFMTOS_23	Number of TCP connections with SYN packets where source IP=current source IP in a 5 second window.
429	DFMTOS_24	Number of TCP connections with SYN packets where destination IP=current source IP in a 5 second window.
430	DFMTOS_25	Number of TCP connections with RST packets where source IP=current source IP in a 5 second window.
431	DFMTOS_26	Number of TCP connections with RST packets where destination IP=current source IP in a 5 second window.
432	DFMTOS_27	Number of TCP packets with destination IP=current source IP in a 5 second window.
433	DFMTOS_28	Number of TCP packets with source IP=current source IP in a 5 second window.
434	DFMTOS_29	Total bytes of TCP packets with destination IP=current source IP in a 5 second window.
435	DFMTOS_30	Total bytes of TCP packets with source IP=current source IP in a 5 second window.
436	DFMTOS_30a	Average (DFMCOS_29/DFMCOS_27) TCP bytes of packets with destination IP=current source IP as the current connection in a 5 second window.
437	DFMTOS_30b	Average (DFMCOS_30/DFMCOS_28) TCP bytes of packets with source IP=current source IP as the current connection in a 5 second window.
438	DFMTOS_31	Number of UDP packets with destination IP=current source IP in a 5 second window.

No.	Feature	Description
439	DFMTOS_32	Number of UDP packets with source IP=current source IP in a 5 second window.
440	DFMTOS_33	Total bytes of UDP packets with destination IP=current source IP in a 5 second window.
441	DFMTOS_34	Total bytes of UDP packets with source IP=current source IP in a 5 second window.
442	DFMTOS_34a	Average (DFMCOS_33/DFMCOS_31) UDP bytes of packets with destination IP=current source IP as the current connection in a 5 second window.
443	DFMTOS_34b	Average (DFMCOS_34/DFMCOS_32) UDP bytes of packets with source IP=current source IP as the current connection in a 5 second window.
444	DFMTOS_35	Number of ICMP packets with destination IP=current source IP in a 5 second window.
445	DFMTOS_36	Number of ICMP packets with source IP=current source IP in a 5 second window.
446	DFMTOS_37	Total bytes of ICMP packets with destination IP=current source IP in a 5 second window.
447	DFMTOS_38	Total bytes of ICMP packets with source IP=current source IP in a 5 second window.
448	DFMTOS_38a	Average (DFMCOS_37/DFMCOS_35) ICMP bytes of packets with destination IP=current source IP as the current connection in a 5 second window.
449	DFMTOS_38b	Average (DFMCOS_38/DFMCOS_36) ICMP bytes of packets with source IP=current source IP as the current connection in a 5 second window.
450	DFMTOS_39	Number of Destination Unreachable (Type 3) ICMP packets with destination IP=current source IP in a 5 second window.
450a	DFMTOS_40	<i>Number of Echo Reply (Type 0) ICMP packets with source IP=current source IP in a 5 second window.</i>
450b	DFMTOS_41	<i>Number of TCP (URG) packets with destination IP=current source IP as the current connection in a 5 second window.</i>
451	DFMTOS_42	Number of TCP (ACK) packets with destination IP=current source IP as the current connection in a 5 second window.
452	DFMTOS_43	Number of TCP (PSH) packets with destination IP=current source IP as the current connection in a 5 second window.
453	DFMTOS_44	Number of TCP (RST) packets with destination IP=current source IP as the current connection in a 5 second window.
454	DFMTOS_45	Number of TCP (SYN) packets with destination IP=current source IP as the current connection in a 5 second window.
455	DFMTOS_46	Number of TCP (FIN) packets with destination IP=current source IP as the current connection in a 5 second window.
455a	DFMTOS_47	<i>Number of TCP (URG) packets with source IP=current source IP as the current connection in a 5 second window.</i>
456	DFMTOS_48	Number of TCP (ACK) packets with source IP=current source IP as the current connection in a 5 second window.
457	DFMTOS_49	Number of TCP (PSH) packets with source IP=current source IP as the current connection in a 5 second window.
458	DFMTOS_50	Number of TCP (RST) packets with source IP=current source IP as the current connection in a 5 second window.
459	DFMTOS_51	Number of TCP (SYN) packets with source IP=current source IP as the current connection in a 5 second window.
460	DFMTOS_52	Number of TCP (FIN) packets with source IP=current source IP as the current connection in a 5 second window.
461	DFMTOD_1	Number of TCP connections with source IP=current destination IP in a 5 second window.
462	DFMTOD_2	Number of UDP connections with source IP=current destination IP in a 5 second window.

No.	Feature	Description
463	DFMTOD_3	Number of ICMP connections with source IP=current destination IP in a 5 second window.
464	DFMTOD_4	Number of TCP connections with destination IP=current destination IP in a 5 second window.
465	DFMTOD_5	Number of UDP connections with destination IP=current destination IP in a 5 second window.
466	DFMTOD_6	Number of ICMP connections with destination IP=current destination IP in a 5 second window.
467	DFMTOD_7	Number of TCP connections with source IP=current destination IP and destination port=current destination port in a 5 second window.
468	DFMTOD_8	Number of TCP connections with source IP=current destination IP and destination port!=current destination port in a 5 second window.
469	DFMTOD_9	Number of TCP connections with destination IP=current destination IP and destination port=current destination port in a 5 second window.
470	DFMTOD_10	Number of TCP connections with destination IP=current destination IP and destination port!=current destination port in a 5 second window.
471	DFMTOD_11	Number of TCP connections with source IP=current destination IP and destination port=current source port in a 5 second window.
472	DFMTOD_12	Number of TCP connections with source IP=current destination IP and destination port!=current source port in a 5 second window.
473	DFMTOD_13	Number of TCP connections with destination IP=current destination IP and destination port=current source port in a 5 second window.
474	DFMTOD_14	Number of TCP connections with destination IP=current destination IP and destination port!=current source port in a 5 second window.
475	DFMTOD_15	Number of UDP connections with source IP=current destination IP and destination port=current destination port in a 5 second window.
476	DFMTOD_16	Number of UDP connections with source IP=current destination IP and destination port!=current destination port in a 5 second window.
477	DFMTOD_17	Number of UDP connections with destination IP=current destination IP and destination port=current destination port in a 5 second window.
478	DFMTOD_18	Number of UDP connections with destination IP=current destination IP and destination port!=current destination port in a 5 second window.
479	DFMTOD_19	Number of UDP connections with source IP=current destination IP and destination port=current source port in a 5 second window.
480	DFMTOD_20	Number of UDP connections with source IP=current destination IP and destination port!=current source port in a 5 second window.
481	DFMTOD_21	Number of UDP connections with destination IP=current destination IP and destination port=current source port in a 5 second window.
482	DFMTOD_22	Number of UDP connections with destination IP=current destination IP and destination port!=current source port in a 5 second window.
483	DFMTOD_23	Number of TCP connections with SYN packets where source IP=current destination IP in a 5 second window.

No.	Feature	Description
484	DFMTOD_24	Number of TCP connections with SYN packets where destination IP=current destination IP in a 5 second window.
485	DFMTOD_25	Number of TCP connections with RST packets where source IP=current destination IP in a 5 second window.
486	DFMTOD_26	Number of TCP connections with RST packets where destination IP=current destination IP in a 5 second window.
487	DFMTOD_27	Number of TCP packets with destination IP=current destination IP in a 5 second window.
488	DFMTOD_28	Number of TCP packets with source IP=current destination IP in a 5 second window.
489	DFMTOD_29	Total bytes of TCP packets with destination IP=current destination IP in a 5 second window.
490	DFMTOD_30	Total bytes of TCP packets with source IP=current destination IP in a 5 second window.
491	DFMTOD_30a	Average (DFMCOS_29/DFMCOS_27) TCP bytes of packets with destination IP=current destination IP as the current connection in a 5 second window.
492	DFMTOD_30b	Average (DFMCOS_30/DFMCOS_28) TCP bytes of packets with source IP=current destination IP as the current connection in a 5 second window.
493	DFMTOD_31	Number of UDP packets with destination IP=current destination IP in a 5 second window.
494	DFMTOD_32	Number of UDP packets with source IP=current destination IP in a 5 second window.
495	DFMTOD_33	Total bytes of UDP packets with destination IP=current destination IP in a 5 second window.
496	DFMTOD_34	Total bytes of UDP packets with source IP=current destination IP in a 5 second window.
497	DFMTOD_34a	Average (DFMCOS_33/DFMCOS_31) UDP bytes of packets with destination IP=current destination IP as the current connection in a 5 second window.
498	DFMTOD_34b	Average (DFMCOS_34/DFMCOS_32) UDP bytes of packets with source IP=current destination IP as the current connection in a 5 second window.
499	DFMTOD_35	Number of ICMP packets with destination IP=current destination IP in a 5 second window.
500	DFMTOD_36	Number of ICMP packets with source IP=current destination IP in a 5 second window.
501	DFMTOD_37	Total bytes of ICMP packets with destination IP=current destination IP in a 5 second window.
502	DFMTOD_38	Total bytes of ICMP packets with source IP=current destination IP in a 5 second window.
503	DFMTOD_38a	Average (DFMCOS_37/DFMCOS_35) ICMP bytes of packets with destination IP=current destination IP as the current connection in a 5 second window.
504	DFMTOD_38b	Average (DFMCOS_38/DFMCOS_36) ICMP bytes of packets with source IP=current destination IP as the current connection in a 5 second window.
505	DFMTOD_39	Number of Destination Unreachable (Type 3) ICMP packets with destination IP=current destination IP in a 5 second window.
505a	<i>DFMTOD_40</i>	<i>Number of Echo Reply (Type 0) ICMP packets with source IP=current destination IP in a 5 second window.</i>
505b	<i>DFMTOD_41</i>	<i>Number of TCP (URG) packets with destination IP=current destination IP as the current connection in a 5 second window.</i>
506	DFMTOD_42	Number of TCP (ACK) packets with destination IP=current destination IP as the current connection in a 5 second window.

No.	Feature	Description
507	DFMTOD_43	Number of TCP (PSH) packets with destination IP=current destination IP as the current connection in a 5 second window.
508	DFMTOD_44	Number of TCP (RST) packets with destination IP=current destination IP as the current connection in a 5 second window.
509	DFMTOD_45	Number of TCP (SYN) packets with destination IP=current destination IP as the current connection in a 5 second window.
510	DFMTOD_46	Number of TCP (FIN) packets with destination IP=current destination IP as the current connection in a 5 second window.
510a	DFMTOD_47	<i>Number of TCP (URG) packets with source IP=current destination IP as the current connection in a 5 second window.</i>
511	DFMTOD_48	Number of TCP (ACK) packets with source IP=current destination IP as the current connection in a 5 second window.
512	DFMTOD_49	Number of TCP (PSH) packets with source IP=current destination IP as the current connection in a 5 second window.
513	DFMTOD_50	Number of TCP (RST) packets with source IP=current destination IP as the current connection in a 5 second window.
514	DFMTOD_51	Number of TCP (SYN) packets with source IP=current destination IP as the current connection in a 5 second window.
515	DFMTOD_52	Number of TCP (FIN) packets with source IP=current destination IP as the current connection in a 5 second window.
516	DFMTG_1	Total number of TCP connections in a 5 second window.
517	DFMTG_2	Total number of UDP connections in a 5 second window.
518	DFMTG_3	Total number of ICMP connections in a 5 second window.
519	DFMTG_4	Number of TCP connections with source or destination port=current destination port in a 5 second window.
520	DFMTG_5	Number of TCP connections with source or destination port!=current destination port in a 5 second window.
521	DFMTG_6	Number of UDP connections with source or destination port=current destination port in a 5 second window.
522	DFMTG_7	Number of UDP connections with source or destination port!=current destination port in a 5 second window.
523	DFMTG_8	Total number of exchanged TCP packets in a 5 second window.
524	DFMTG_9	Total number of exchanged UDP packets in a 5 second window.
525	DFMTG_10	Total number of exchanged ICMP packets in a 5 second window.
526	DFMTG_11	Number of TCP connections with destination port=current source port in a 5 second window.
527	DFMTG_12	Number of UDP connections with destination port=current source port in a 5 second window.
528	DFMTG_13	Number of TCP connections with source port=current source port in a 5 second window.
529	DFMTG_14	Number of UDP connections with source port=current source port in a 5 second window.
530	DFMTG_15	Number of TCP connections with destination port=current destination port in a 5 second window.
531	DFMTG_16	Number of UDP connections with destination port=current destination port in a 5 second window.
532	DFMTG_17	Number of TCP connections with source port=current destination port in a 5 second window.
533	DFMTG_18	Number of UDP connections with source port=current destination port in a 5 second window.
533a	DFMTG_19	<i>Number of TCP (URG) packets in a 5 second window.</i>
534	DFMTG_20	Number of TCP (ACK) packets in a 5 second window.
535	DFMTG_21	Number of TCP (PSH) packets in a 5 second window.

No.	Feature	Description
536	DFMTG_22	Number of TCP (RST) packets in a 5 second window.
537	DFMTG_23	Number of TCP (SYN) packets in a 5 second window.
538	DFMTG_24	Number of TCP (FIN) packets in a 5 second window.
539	DFMTG_25	Number of Destination Unreachable (Type 3) ICMP packets in a 5 second window.
539a	DFMTG_26	<i>Number of Echo Reply (Type 0) ICMP packets in a 5 second window.</i>
539b	DFMTG_27	<i>Number of Destination Unreachable (Type 3) ICMP connections in a 5 second window.</i>
539c	DFMTG_28	<i>Number of Echo Reply (Type 0) ICMP connections in a 5 second window.</i>
540	DFMTG_29	Number of TCP connection with SYN packets and destination port=current source port in a 5 second window.
541	DFMTG_30	Number of TCP connection with SYN packets and source port=current source port in a 5 second window.
542	DFMTG_31	Number of TCP connection with SYN packets and destination port=current destination port in a 5 second window.
543	DFMTG_32	Number of TCP connection with SYN packets and source port=current destination port in a 5 second window.
544	DFMTG_33	Number of TCP connection with RST packets and destination port=current source port in a 5 second window.
545	DFMTG_34	Number of TCP connection with RST packets and source port=current source port in a 5 second window.
546	DFMTG_35	Number of TCP connection with RST packets and destination port=current destination port in a 5 second window.
547	DFMTG_36	Number of TCP connection with RST packets and source port=current destination port in a 5 second window.
548	DFMTG_37	Total number of TCP connections with SYN packets in a 5 second window.
549	DFMTG_38	Total number of TCP connections with RST packets in a 5 second window.

D.4. Class Feature

No.	Feature	Description
550	class	Connection label {Normal, Attack}

REFERENCES

- [1] Amjad Al Tobi and Ishbel Duncan. KDD 1999 Generation Faults: A Review and Analysis. *Journal of Cyber Security Technology*, 2018. URL <https://doi.org/10.1080/23742917.2018.1518061>.
- [2] Yulia Cherdantseva and Jeremy Hilton. A Reference Model of Information Assurance & Security. In *2013 International Conference on Availability, Reliability and Security*, pages 546–555. IEEE, 2013.
- [3] Dave Gordon. *Managing Auditability (and Other Non-Functional, Non-Technical Requirements)*, 2014. URL <http://searchsecurity.techtarget.com/definition/nonrepudiation>, Accessed 16 Jul 2018.
- [4] Chad Perrin. *The CIA Triad*, 2008. URL <https://www.techrepublic.com/blog/it-security/the-cia-triad/>, Accessed 16 Jul 2018.
- [5] Margaret Rouse. *Nonrepudiation*, 2008. URL <http://searchsecurity.techtarget.com/definition/nonrepudiation>, Accessed 16 Jul 2018.
- [6] Hossein Bidgoli. *Handbook of Information Security, Information Warfare, Social, Legal, and International Issues and Security Foundations*, volume 2. John Wiley & Sons, 2006.
- [7] Steven Andrew Hofmeyr. *An Immunological Model of Distributed Detection and Its Application to Computer Security*. PhD thesis, University of New Mexico, Department of Computer Science, 1999.
- [8] Swagatam Das, Shounak Datta, and Bidyut B Chaudhuri. Handling Data Irregularities in Classification: Foundations, Trends, and Future Challenges. *Pattern Recognition*, pages 674–693, 2018.
- [9] Sotiris B Kotsiantis. Supervised Machine Learning: A Review of Classification Techniques. *Informatica (Ljubljana)*, 31 (3): 249–268, 2007.
- [10] Nikolay Burlutskiy, Miltos Petridis, Andrew Fish, Alexey Chernov, and Nour Ali. An Investigation on Online Versus Batch Learning in Predicting User Behaviour. In *Research and Development in Intelligent Systems XXXIII*, pages 135–149. Springer International Publishing, 2016.
- [11] Dorothy E Denning. An Intrusion-Detection Model. *IEEE Transactions on Software Engineering*, SE-13 (2): 222–232, 1987.
- [12] Monowar H Bhuyan, Dhruva Kumar Bhattacharyya, and Jugal K Kalita. Network Anomaly Detection: Methods, Systems and Tools. *IEEE Communications Surveys & Tutorials*, 16 (1): 303–336, 2014.
- [13] Stefan Axelsson. The Base-Rate Fallacy and the Difficulty of Intrusion Detection. *ACM Transactions on Information and System Security (TISSEC)*, 3 (3): 186–205, 2000.

- [14] Ali Shiravi, Hadi Shiravi, Mahbod Tavallaee, and Ali A Ghorbani. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *Computers & Security*, 31 (3): 357–374, 2012.
- [15] Iosif-Viorel Onut and Ali A Ghorbani. A Feature Classification Scheme for Network Intrusion Detection. *IJ Network Security*, 5 (1): 1–15, 2007.
- [16] Vern Paxson. Bro: a system for detecting network intruders in real-time. *Computer Networks*, 31 (23-24): 2435–2463, 1999.
- [17] Janez Demšar. Statistical Comparisons of Classifiers over Multiple Data Sets. *Journal of Machine learning research*, 7 (Jan): 1–30, 2006.
- [18] James P Anderson. Computer Security Technology Planning Study (Vol. I). Technical report, Anderson (James P) and Co Fort Washington PA, 1972.
- [19] James P Anderson. Computer Security Technology Planning Study (Vol. II). Technical report, Anderson (James P) and Co Fort Washington PA, 1972.
- [20] James P Anderson. Computer Security Threat Monitoring and Surveillance. Technical report, Anderson (James P) and Co Fort Washington PA, 1980.
- [21] Emmanouil Vasilomanolakis, Shankar Karuppayah, Max Mühlhäuser, and Mathias Fischer. Taxonomy and Survey of Collaborative Intrusion Detection. *ACM Computing Surveys (CSUR)*, 47 (4): 55:1–55:33, 2015.
- [22] Tarfa Hamed, Jason B Ernst, and Stefan C Kremer. A Survey and Taxonomy on Data and Pre-processing Techniques of Intrusion Detection Systems. In *Computer and Network Security Essentials*, pages 113–134. Springer, 2018.
- [23] Reema Patel, Amit Thakkar, and Amit Ganatra. A Survey and Comparative Analysis of Data Mining Techniques for Network Intrusion Detection Systems. *International Journal of Soft Computing and Engineering (IJSCE) ISSN*, pages 2231–2307, 2012.
- [24] Robert Mitchell and Ing-Ray Chen. A Survey of Intrusion Detection Techniques for Cyber-Physical Systems. *ACM Computing Surveys (CSUR)*, 46 (4): 55:1–55:29, 2014.
- [25] Hung-Jen Liao, Chun-Hung Richard Lin, Ying-Chih Lin, and Kuang-Yuan Tung. Intrusion Detection System: A Comprehensive Review. *Journal of Network and Computer Applications*, 36 (1): 16–24, 2013.
- [26] Iginio Corona, Giorgio Giacinto, and Fabio Roli. Adversarial Attacks Against Intrusion Detection Systems: Taxonomy, Solutions and Open Issues. *Information Sciences*, 239: 201–225, 2013.
- [27] Anna L Buczak and Erhan Guven. A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. *IEEE Communications Surveys & Tutorials*, 18 (2): 1153–1176, 2016.
- [28] Chirag Modi, Dhiren Patel, Bhavesh Borisaniya, Hiren Patel, Avi Patel, and Muttukrishnan Rajarajan. A Survey of Intrusion Detection Techniques in Cloud. *Journal of Network and Computer Applications*, 36 (1): 42–57, 2013.
- [29] Richard Zuech, Taghi M Khoshgoftaar, and Randall Wald. Intrusion Detection and Big Heterogeneous Data: A Survey. *Journal of Big Data*, 2 (1): 1–14, 2015.
- [30] Jason Nikolai and Yong Wang. Hypervisor-Based Cloud Intrusion Detection System. In *proceedings of the International Conference on Computing, Networking and Communications (ICNC)*, pages 989–993. IEEE, 2014.

- [31] Carlos A Catania and Carlos García Garino. Automatic Network Intrusion Detection: Current Techniques and Open Issues. *Computers & Electrical Engineering*, 38 (5): 1062–1072, 2012.
- [32] Ali A Ghorbani, Wei Lu, and Mahbod Tavallaee. *Network Intrusion Detection and Prevention: Concepts and Techniques*, volume 47. Springer Science & Business Media, 2010.
- [33] Aurobindo Sundaram. An Introduction to Intrusion Detection. *Crossroads*, 2 (4): 3–7, 1996.
- [34] Byung-Chul Park, Young J Won, Myung-Sup Kim, and James W Hong. Towards Automated Application Signature Generation for Traffic Identification. In *Proceedings of the Network Operations and Management Symposium (NOMS)*, pages 160–167. IEEE, 2008.
- [35] Matthew Van Gundy, Davide Balzarotti, and Giovanni Vigna. Catch Me, If You Can: Evading Network Signatures with Web-based Polymorphic Worms. In *proceedings of the first USENIX Workshop on Offensive Technologies (WOOT)*, pages 7:1–7:9, 2007.
- [36] Christopher Kruegel, William Robertson, and Giovanni Vigna. Using Alert Verification to Identify Successful Intrusion Attempts. *Praxis der Informationsverarbeitung und Kommunikation*, 27 (4): 219–227, 2004.
- [37] Damiano Bolzoni, Bruno Crispo, and Sandro Etalle. ATLANTIDES: An Architecture for Alert Verification in Network Intrusion Detection Systems. In *proceedings of the 21st Conference on Large Installation System Administration Conference (LISA)*, volume 7, pages 1–12, 2007.
- [38] Kevin Borders, Xin Zhao, and Atul Prakash. Siren: Catching Evasive Malware. In *proceedings of the IEEE Symposium on Security and Privacy*, pages 1–6. IEEE, 2006.
- [39] Roberto Perdisci, Guofei Gu, and Wenke Lee. Using an Ensemble of One-Class SVM Classifiers to Harden Payload-based Anomaly Detection Systems. In *proceedings of the 6th International Conference on Data Mining (ICDM)*, pages 488–498. IEEE, 2006.
- [40] Xavier JA Bellekens, Christos Tachtatzis, Robert C Atkinson, Craig Renfrew, and Tony Kirkham. A Highly-Efficient Memory-Compression Scheme for GPU-Accelerated Intrusion Detection Systems. In *proceedings of the 7th International Conference on Security of Information and Networks*, pages 302–309. ACM, 2014.
- [41] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly Detection: A Survey. *ACM Computing Surveys (CSUR)*, 41 (3): 5, 2009.
- [42] Wei Lu and Ali A Ghorbani. Network Anomaly Detection Based on Wavelet Analysis. *EURASIP Journal on Advances in Signal Processing*, 2009: 4:1–4:16, 2009.
- [43] Stuart Staniford, James A Hoagland, and Joseph M McAlerney. Practical Automated Detection of Stealthy Portscans. *Journal of Computer Security*, 10 (1-2): 105–136, 2002.
- [44] Zheng Zhang, Jun Li, CN Manikopoulos, Jay Jorgenson, and Jose Ucles. HIDE: A Hierarchical Network Intrusion Detection System Using Statistical Preprocessing and Neural Network Classification. In *proceedings of the IEEE Workshop on Information Assurance and Security*, pages 85–90, 2001.

- [45] Matthew V Mahoney and Philip K Chan. PHAD: Packet Header Anomaly Detection for Identifying Hostile Network Traffic. Technical report, 2001.
- [46] Ke Wang and Salvatore J Stolfo. Anomalous Payload-Based Network Intrusion Detection. In *International Workshop on Recent Advances in Intrusion Detection*, volume 3224, pages 203–222. Springer, 2004.
- [47] Matthew V Mahoney and Philip K Chan. Learning Rules for Anomaly Detection of Hostile Network Traffic. In *proceedings of the 3rd IEEE International Conference on Data Mining (ICDM)*, pages 601–604. IEEE, 2003.
- [48] Sui Song, Li Ling, and CN Manikopoulo. Flow-Based Statistical Aggregation Schemes for Network Anomaly Detection. In *proceedings of the 2006 IEEE International Conference on Networking, Sensing and Control (ICNSC)*, pages 786–791. IEEE, 2006.
- [49] Shi-Jinn Horng, Ming-Yang Su, Yuan-Hsin Chen, Tzong-Wann Kao, Rong-Jian Chen, Jui-Lin Lai, and Citra Dwi Perkasa. A Novel Intrusion Detection System Based On Hierarchical Clustering and Support Vector Machines. *Expert systems with Applications*, 38 (1): 306–313, 2011.
- [50] Zaniah Muda, Warusia Yassin, Md. Nasir Sulaiman, and Nur Izura Udzir. Intrusion Detection based on K-Means Clustering and Naïve Bayes Classification. In *proceedings of the 7th International Conference on Information Technology in Asia (CITA)*, pages 1–6. IEEE, 2011.
- [51] Anteneh Girma, Mosses Garuba, and Rajini Goel. Advanced Machine Language Approach to Detect DDoS Attack Using DBSCAN Clustering Technology with Entropy. In *proceedings of the Information Technology - New Generations. Advances in Intelligent Systems and Computing*, volume 558, pages 125–131. Springer International Publishing, 2018.
- [52] Gilbert R Hendry and Shanchieh J Yang. Intrusion Signature Creation via Clustering Anomalies. In *proceedings of the Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security*, volume 6973, pages 1–12. International Society for Optics and Photonics, 2008.
- [53] Ira Assent. Clustering High Dimensional Data. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2 (4): 340–350, 2012.
- [54] Shweta Malhotra, Vikram Bali, and KK Paliwal. Genetic Programming and K-Nearest Neighbour Classifier Based Intrusion Detection Model. In *proceedings of the 7th International Conference on Cloud Computing, Data Science & Engineering-Confluence*, pages 42–46. IEEE, 2017.
- [55] Imen Boukhris, Zied Elouedi, and Mariem Ajabi. Toward Intrusion Detection Using Belief Decision Trees for Big Data. *Knowledge and Information Systems*, 53 (3): 671–698, 2017.
- [56] R Vijayanand, D Devaraj, and B Kannapiran. Intrusion detection system for wireless mesh network using multiple support vector machine classifiers with genetic-algorithm-based feature selection. *Computers & Security*, 77: 304–314, 2018.
- [57] Shawq Malik Mehibs and Soukaena Hassan Hashim. Proposed Network Intrusion Detection System in Cloud Environment Based on Back Propagation Neural Network. *Journal of University of Babylon*, 26 (1): 29–40, 2018.

- [58] Saurabh Mukherjee and Neelam Sharma. Intrusion Detection using Naive Bayes Classifier with Feature Reduction . *Procedia Technology*, 4: 119–128, 2012.
- [59] William W Cohen. Fast Effective Rule Induction. In *proceedings of the 12th International Conference on Machine Learning*, pages 115–123. Morgan Kaufmann, 1995.
- [60] Daniel Barbará, Julia Couto, Sushil Jajodia, and Ningning Wu. ADAM: A Testbed for Exploring the Use of Data Mining in Intrusion Detection. *ACM Sigmod Record*, 30 (4): 15–24, 2001.
- [61] Latifur Khan, Mamoun Awad, and Bhavani Thuraisingham. A New Intrusion Detection System Using Support Vector Machines and Hierarchical Clustering. *The International Journal on Very Large Data Bases (VLDB)*, 16 (4): 507–521, 2007.
- [62] Michael E Locasto, Ke Wang, Angelos D Keromytis, and Salvatore J Stolfo. Flips: Hybrid Adaptive Intrusion Prevention. In *International Workshop on Recent Advances in Intrusion Detection*, pages 82–101. Springer, 2005.
- [63] Jiong Zhang and Mohammad Zulkernine. A Hybrid Network Intrusion Detection Technique Using Random Forests. In *proceedings of the First International Conference on Availability, Reliability and Security (ARES)*, pages 1–8. IEEE, 2006.
- [64] Sandhya Peddabachigari, Ajith Abraham, Crina Grosan, and Johnson Thomas. Modeling intrusion detection system using hybrid intelligent systems. *Journal of Network and Computer Applications*, 30 (1): 114–132, 2007.
- [65] Jiong Zhang, Mohammad Zulkernine, and Anwar Haque. Random-Forests-Based Network Intrusion Detection Systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38 (5): 649–659, 2008.
- [66] M Ali Aaydn, A Halim Zaim, and K Gökhan Ceylan. A Hybrid Intrusion Detection System Design for Computer Network Security. *Computers & Electrical Engineering*, 35 (3): 517–526, 2009.
- [67] Sahar Selim, Mohamed Hashem, and Taymoor M Nazmy. Hybrid Multi-Level Intrusion Detection System. *International Journal of Computer Science and Information Security*, 9 (5): 23–29, 2011.
- [68] Mrutyunjaya Panda, Ajith Abraham, and Manas Ranjan Patra. Hybrid Intelligent Systems for Detecting Network Intrusions. *Security and Communication Networks*, 8 (16): 2741–2749, 2012. URL <http://dx.doi.org/10.1002/sec.592>.
- [69] Xiaojun Tong, Zhu Wang, and Haining Yu. A research using hybrid RBF/Elman neural networks for intrusion detection system secure model. *Computer Physics Communications*, 180 (10): 1795–1801, 2009.
- [70] Xuedou Yu. A New Model of Intelligent Hybrid Network Intrusion Detection System. In *proceedings of the International Conference on Bioinformatics and Biomedical Technology (ICBBT)*, pages 386–389. IEEE, 2010.
- [71] Álvaro Herrero, Martí Navarro, Emilio Corchado, and Vicente Julián. RT-MOVICAB-IDS: Addressing real-time intrusion detection. *Future Generation Computer Systems*, 29 (1): 250–261, 2013.
- [72] N Subramanian, Pramod S Pawar, Mayank Bhatnagar, Nihar S Khedekar, Srinivas Guntupalli, N Satyanarayana, VK Vijaykumar, Praveen K Ampatt, Rajiv Ranjan, and Prasad J Pandit. Development of a Comprehensive Intrusion Detection System -

- Challenges and Approaches. In *International Conference on Information Systems Security*, pages 332–335. Springer, 2005.
- [73] Steven R Snapp, James Brentano, Gihan V Dias, Terrance L Goan, L Todd Heberlein, Che-Lin Ho, Karl N Levitt, Biswanath Mukherjee, Stephen E Smaha, Tim Grance, Daniel M. Teal, and Doug Mansur. DIDS (Distributed Intrusion Detection System)-Motivation, Architecture, and an Early Prototype. In *proceedings of the 14th National Computer Security Conference*, volume 1, pages 167–176. Washington, DC, 1991.
- [74] Paul Baecher, Markus Koetter, Thorsten Holz, Maximillian Dornseif, and Felix Freiling. The Nepenthes Platform: An Efficient Approach to Collect Malware. In *International Workshop on Recent Advances in Intrusion Detection*, pages 165–184. Springer, 2006.
- [75] Georgios Portokalidis, Asia Slowinska, and Herbert Bos. Argos: An Emulator for Fingerprinting Zero-Day Attacks for Advertised Honeypots with Automatic Signature Generation. In *ACM SIGOPS Operating Systems Review*, volume 40, pages 15–27. ACM, 2006.
- [76] Herve Debar, David A Curry, and Benjamin S Feinstein. The Intrusion Detection Message Exchange Format (IDMEF). RFC 4765, RFC Editor, 2007. URL <https://www.rfc-editor.org/info/rfc4765>, Accessed 28 Aug 2018.
- [77] Stuart Staniford-Chen, Steven Cheung, Richard Crawford, Mark Dilger, Jeremy Frank, James Hoagland, Karl Levitt, Christopher Wee, Raymond Yip, and Dan Zerkle. GrIDS -A Graph Based Intrusion Detection System for Large Networks. In *proceedings of the 19th National Information Systems Security Conference*, volume 1, pages 361–370. Baltimore, 1996.
- [78] Steven Cheung, Rick Crawford, Mark Dilger, Jeremy Frank, Jim Hoagland, Karl Levitt, Jeff Rowe, Stuart Staniford-Chen, Raymond Yip, and Dan Zerkle. The Design of GrIDS: A Graph-Based Intrusion Detection System. In *proceedings of the 19th National Information Systems Security Conference*, pages 361–370, 1999.
- [79] Phillip A Porras and Peter G Neumann. EMERALD: Event Monitoring Enabling Response to Anomalous Live Disturbances. In *proceedings of the 20th National Information Systems Security Conference*, pages 353–365, 1997.
- [80] Chi-Chun Lo, Chun-Chieh Huang, and Joy Ku. A Cooperative Intrusion Detection System Framework for Cloud Computing Networks. In *proceedings of the 39th International Conference on Parallel Processing Workshops (ICPPW)*, pages 280–284. IEEE, 2010.
- [81] Amir Vahid Dastjerdi, Kamalrulnizam Abu Bakar, and Sayed Gholam Hassan Tabatabaei. Distributed Intrusion Detection in Clouds Using Mobile Agents. In *proceedings of the 3rd International Conference on Advanced Engineering Computing and Applications in Sciences*, pages 175–180. IEEE, 2009.
- [82] M Sanjay Ram. Secure Cloud Computing Based on Mutual Intrusion Detection System. *International Journal of Computer Application*, 1 (2): 57–67, 2012.
- [83] Chenfeng Vincent Zhou, Shanika Karunasekera, and Christopher Leckie. Evaluation of a Decentralized Architecture for Large Scale Collaborative Intrusion Detection. In *proceedings of the 10th IFIP/IEEE International Symposium on Integrated Network Management*, pages 80–89. IEEE, 2007.

- [84] Chenfeng Vincent Zhou, Shanika Karunasekera, and Christopher Leckie. A Peer-to-Peer Collaborative Intrusion Detection System. In *13th IEEE International Conference on Networks, 2005. Jointly held with the 2005 IEEE 7th Malaysia International Conference on Communication.*, volume 1, pages 118–123. IEEE, 2005.
- [85] Sean Rhea, Dennis Geels, Timothy Roscoe, and John Kubiawicz. Handling Churn in a DHT. In *proceedings of the Annual Conference on USENIX Annual Technical Conference, 2003*.
- [86] Sean Rhea, Brighten Godfrey, Brad Karp, John Kubiawicz, Sylvia Ratnasamy, Scott Shenker, Ion Stoica, and Harlan Yu. OpenDHT: A Public DHT Service and Its Uses. In *ACM SIGCOMM Computer Communication Review*, volume 35, pages 73–84. ACM, 2005.
- [87] Vinod Yegneswaran, Paul Barford, and Somesh Jha. Global Intrusion Detection in the DOMINO Overlay System. In *proceedings of the Network and Distributed Systems Security Symposium (NDSS), 2004*.
- [88] Tal Garfinkel and Mendel Rosenblum. A Virtual Machine Introspection Based Architecture for Intrusion Detection. In *proceedings of the Network and Distributed Systems Security Symposium (NDSS)*, volume 3, pages 191–206, 2003.
- [89] Sin Yeung Lee, Wai Lup Low, and Pei Yuen Wong. Learning Fingerprints for a Database Intrusion Detection System. In *European Symposium on Research in Computer Security*, pages 264–279. Springer, 2002.
- [90] Jong Chun Park and Jedidiah R Crandall. Empirical Study of a National-Scale Distributed Intrusion Detection System: Backbone-Level Filtering of HTML Responses in China. In *proceedings of the IEEE 30th International Conference on Distributed Computing Systems (ICDCS)*, pages 315–326. IEEE, 2010.
- [91] R Sekar, Ajay Gupta, James Frullo, Tushar Shanbhag, Abhishek Tiwari, Henglin Yang, and Sheng Zhou. Specification-Based Anomaly Detection: A New Approach for Detecting Network Intrusions. In *proceedings of the 9th ACM Conference on Computer and Communications Security*, pages 265–274. ACM, 2002.
- [92] Taeshik Shon and Jongsub Moon. A Hybrid Machine Learning Approach to Network Anomaly Detection. *Information Sciences*, 177 (18): 3799–3821, 2007.
- [93] Xiaohui Jin, Baojiang Cui, Dong Li, Zishuai Cheng, and Congxian Yin. An Improved Payload-Based Anomaly Detector for Web Applications. *Journal of Network and Computer Applications*, pages 111–116, 2018.
- [94] Roberto Perdisci, Davide Ariu, Prahlad Fogla, Giorgio Giacinto, and Wenke Lee. McPAD: A multiple classifier system for accurate payload-based anomaly detection. *Computer Networks*, 53 (6): 864–881, 2009.
- [95] Paulo M Mafra, Vinicius Moll, Joni da Silva Fraga, and Altair Olivo Santin. Octopus-IIDS: An Anomaly Based Intelligent Intrusion Detection System. In *proceedings of the IEEE symposium on Computers and Communications (ISCC)*, pages 405–410. IEEE, 2010.
- [96] Srilatha Chebrolu, Ajith Abraham, and Johnson P Thomas. Feature Deduction and Ensemble Design of Intrusion Detection Systems. *Computers & security*, 24 (4): 295–307, 2005.

- [97] Vrushank Shah, Akshai Aggarwal, and Nirbhay Chaubey. Alert Fusion of Intrusion Detection systems using Fuzzy Dempster shafer Theory. *Journal of Engineering Science and Technology Review*, 10 (3): 123–127, 2017.
- [98] Kalyan Veeramachaneni, Ignacio Araldo, Vamsi Korrapati, Constantinos Bassias, and Ke Li. AI²: Training a big data machine to defend . In *proceedings of the IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS)*, pages 49–54. IEEE, 2016.
- [99] Dimitrios Papamartzivanos, Félix Gómez Mármol, and Georgios Kambourakis. Dendron: Genetic trees driven rule induction for network intrusion detection systems. *Future Generation Computer Systems*, 79: 558–574, 2018.
- [100] Biswapriyo Chakrabarty, Omit Chanda, and Md Saiful Islam. Anomaly based Intrusion Detection System using Genetic Algorithm and K-Centroid Clustering. *International Journal of Computer Applications*, 163 (11): 13–17, 2017.
- [101] B Balajinath and SV Raghavan. Intrusion detection through learning behavior model. *Computer Communications*, 24 (12): 1202–1212, 2001.
- [102] M Sadiq Ali Khan. Rule Based Network Intrusion Detection Using Genetic Algorithm. *International Journal of Computer Applications*, 18 (8): 26–29, 2011.
- [103] Ren Hui Gong, Mohammad Zulkernine, and Purang Abolmaesumi. A Software Implementation of a Genetic Algorithm Based Approach to Network Intrusion Detection. In *proceedings of the 6th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing and First ACIS International Workshop on Self-Assembling Wireless Networks*, pages 246–253. IEEE, 2005.
- [104] Y Dhanalakshmi and I Ramesh Babu. Intrusion Detection Using Data Mining Along Fuzzy Logic and Genetic Algorithms. *International Journal of Computer Science and Network Security*, 8 (2): 27–32, 2008.
- [105] Gail A. Carpenter and Stephen Grossberg. Adaptive Resonance Theory. Technical report, CAS/CNS Technical Reports Series, (008), 2009.
- [106] Teuvo Kohonen. The Self-Organizing Map. *Proceedings of the IEEE*, 78 (9): 1464–1480, 1990.
- [107] Khaled Labib and V Rao Vemuri. NSOM: A Tool to Detect Denial of Service Attacks Using Self-Organizing Maps. *Technical Report*, 2002. URL <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.2.3631>, Accessed 28 Aug 2018.
- [108] Tanujit Chakraborty, Swarup Chattopadhyay, and Ashis Kumar Chakraborty. A novel hybridization of classification trees and artificial neural networks for selection of students in a business school. *Journal of the Operational Research Society of India (OPSEARCH)*, 55 (2): 434–446, 2018.
- [109] Sanjiban Sekhar Roy, Abhinav Mallik, Rishab Gulati, Mohammad S Obaidat, and PV Krishna. A Deep Learning Based Artificial Neural Network Approach for Intrusion Detection. In *International Conference on Mathematics and Computing*, pages 44–53. Springer, 2017.

- [110] Morteza Amini, Rasool Jalili, and Hamid Reza Shahriari. RT-UNNID: A practical solution to real-time network-based intrusion detection using unsupervised neural networks. *Computers & Security*, 25 (6): 459–468, 2006.
- [111] Jianjing Sun, Han Yang, Jingwen Tian, and Fan Wu. Intrusion Detection Method Based on Wavelet Neural Network. In *Knowledge Discovery and Data Mining, 2009. WKDD 2009. Second International Workshop on*, pages 851–854. IEEE, 2009.
- [112] Laheeb Mohammad Ibrahim. Anomaly Network Intrusion Detection System Based on Distributed Time-Delay Neural Network (DTDNN). *Journal of Engineering Science and Technology*, 5 (4): 457–471, 2010.
- [113] Shahram Behzad, Reza Fotohi, Jaber Hosseini Balov, and Mohammad Javad Rabipour. An Artificial Immune Based Approach for Detection and Isolation Misbehavior Attacks in Wireless Networks. *Journal of Computers*, 13 (6): 705–721, 2018.
- [114] Andrea Visconti and Hooman Tahayori. Artificial Immune System Based on Interval Type-2 Fuzzy Set Paradigm. *Applied Soft Computing*, 11 (6): 4055–4063, 2011.
- [115] Yichi Zhang, Lingfeng Wang, Weiqing Sun, Robert C Green, and Mansoor Alam. Artificial Immune System Based Intrusion Detection in a Distributed Hierarchical Network Architecture of Smart Grid. In *Power and Energy Society General Meeting, 2011 IEEE*, pages 1–8. IEEE, 2011.
- [116] Yichi Zhang, Lingfeng Wang, Weiqing Sun, Robert C Green II, and Mansoor Alam. Distributed Intrusion Detection System in A Multi-Layer Network Architecture of Smart Grids. *IEEE Transactions on Smart Grid*, 2 (4): 796–808, 2011.
- [117] S. Shalini, Nihara A. Shafreen, Priya L. Sathiya, and V. Vetriselvi. Intrusion Detection System for Software-Defined Networks Using Fuzzy System. In *Proceedings of the International Conference on Computing and Communication Systems*, pages 603–620. Springer, 2018.
- [118] Salma Elhag, Alberto Fernández, Abdulrahman Altalhi, Saleh Alshomrani, and Francisco Herrera. A multi-objective evolutionary fuzzy system to obtain a broad and accurate set of solutions in intrusion detection systems. *Soft Computing*, pages 1–16, 2017.
- [119] Arman Tajbakhsh, Mohammad Rahmati, and Abdolreza Mirzaei. Intrusion Detection Using Fuzzy Association Rules. *Applied Soft Computing*, 9 (2): 462–469, 2009.
- [120] Ji-Qing Xian, Feng-Hua Lang, and Xian-Lun Tang. A Novel Intrusion Detection Method Based on Clonal Selection Clustering Algorithm. In *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, volume 6, pages 3905–3910. IEEE, 2005.
- [121] Piyakul Tillapart, Thanachai Thumthawatworn, and Pratit Santiprabhob. Fuzzy Intrusion Detection System. *Assumption University Journal of Technology*, 6 (2): 109–114, 2002.
- [122] Ming-Yang Su, Gwo-Jong Yu, and Chun-Yuen Lin. A real-time network intrusion detection system for large-scale attacks based on an incremental mining approach. *Computers & security*, 28 (5): 301–309, 2009.
- [123] Kleber Vieira, Alexandre Schulter, Carlos Westphall, and Carla Westphall. Intrusion Detection for Grid and Cloud Computing. *IT Professional*, 12 (4): 38–43, 2010.

- [124] R Shanmugavadivu and N Nagarajan. Network Intrusion Detection System Using Fuzzy Logic. *Indian Journal of Computer Science and Engineering (IJCSE)*, 2 (1): 101–111, 2011.
- [125] Ru Zhang, Tao Guo, and Jianyi Liu. An IDS Alerts Aggregation Algorithm Based on Rough Set Theory. In *IOP Conference Series: Materials Science and Engineering*, volume 322, pages 1–7. IOP Publishing, 2018.
- [126] Rui-Hong Dong, Dong-Fang Wu, and Qiu-Yu Zhang. The Integrated Artificial Immune Intrusion Detection Model Based on Decision-Theoretic Rough Set. *IJ Network Security*, 19 (6): 880–888, 2017.
- [127] Witcha Chimphlee, Abdul Hanan Abdullah, Mohd Noor Md Sap, Surat Srinoy, and Siriporn Chimphlee. Anomaly-Based Intrusion Detection Using Fuzzy Rough Clustering. In *proceedings of the International Conference on Hybrid Information Technology (ICHIT)*, volume 1, pages 329–334. IEEE, 2006.
- [128] Adebayo O Adetunmbi, Samuel O Falaki, Olumide S Adewale, and Boniface K Alese. Network Intrusion Detection Based on Rough Set and K-Nearest Neighbour. *International Journal of Computing and ICT Research*, 2 (1): 60–66, 2008.
- [129] Rung-Ching Chen, Kai-Fan Cheng, Ying-Hao Chen, and Chia-Fen Hsieh. Using Rough Set and Support Vector Machine for Network Intrusion Detection System. In *proceedings of the First Asian Conference on Intelligent Information and Database Systems (ACIIDS)*, pages 465–470. IEEE, 2009.
- [130] Frans Hendrik Botes, Louise Leenen, and Retha De La Harpe. Ant Colony Induced Decision Trees for Intrusion Detection. In *ECCWS 2017 16th European Conference on Cyber Warfare and Security*, pages 53–62. Academic Conferences and publishing limited, 2017.
- [131] K Kanaka Vardhini and T Sitamahalakshmi. Enhanced Intrusion Detection System Using Data Reduction: An Ant Colony Optimization Approach. *International Journal of Applied Engineering Research*, 12 (9): 1844–1847, 2017.
- [132] Hai-Hua Gao, Hui-Hua Yang, and Xing-Yu Wang. Ant Colony Optimization Based Network Intrusion Feature Selection and Detection. In *proceedings of 2005 International Conference on Machine Learning and Cybernetics*, volume 6, pages 3871–3875. IEEE, 2005.
- [133] Ozgu Can, Murat Osman Unalir, Emine Sezer, Okan Bursa, and Batuhan Erdogdu. An Ontology Based Approach for Host Intrusion Detection Systems. In *Research Conference on Metadata and Semantics Research*, pages 80–86. Springer, 2017.
- [134] Nenekazi Nokuthala Penelope Mkuzangwe and Fulufhelo Vincent Nelwamondo. A Fuzzy Logic Based Network Intrusion Detection System for Predicting the TCP SYN Flooding Attack. In *Asian conference on intelligent information and database systems*, pages 14–22. Springer, 2017.
- [135] Heitor Murilo Gomes, Jean Paul Barddal, Fabrício Enembreck, and Albert Bifet. A Survey on Ensemble Learning for Data Stream Classification. *ACM Computing Surveys (CSUR)*, 50 (2): 23:1–23:36, 2017.
- [136] S Ranjitha Kumari and PK Kumari. Adaptive Anomaly Intrusion Detection System Using Optimized Hoeffding Tree. *ARN Journal of Engineering and Applied Sciences*, 9: 1903–1910, 2014.

- [137] Alexey Tsymbal. The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin*, 106 (2): 1–7, 2004.
- [138] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A Survey on Concept Drift Adaptation. *ACM computing surveys (CSUR)*, 46 (4): 1:1–1:44, 2014.
- [139] Charu C Aggarwal. On Change Diagnosis in Evolving Data Streams. *IEEE Transactions on Knowledge and Data Engineering*, 17 (5): 587–600, 2005.
- [140] Jean Paul Barddal, Heitor Murilo Gomes, and Fabrício Enembreck. A Survey on Feature Drift Adaptation. In *proceedings of the IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 1053–1060. IEEE, 2015.
- [141] Jean Paul Barddal, Heitor Murilo Gomes, Fabrício Enembreck, and Bernhard Pfahringer. A survey on feature drift adaptation: Definition, benchmark, challenges and future directions. *Journal of Systems and Software*, 127: 278–294, 2017.
- [142] Hai-Long Nguyen, Yew-Kwong Woon, Wee-Keong Ng, and Li Wan. Heterogeneous Ensemble for Feature Drifts in Data Streams. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 1–12. Springer, 2012.
- [143] Albert Bifet, Geoff Holmes, Richard Kirkby, and Bernhard Pfahringer. Moa: Massive Online Analysis. *Journal of Machine Learning Research*, 11 (May): 1601–1604, 2010.
- [144] Jeremy Z Kolter and Marcus A Maloof. Using Additive Expert Ensembles to Cope With Concept Drift. In *proceedings of the 22nd International Conference on Machine Learning*, pages 449–456. ACM, 2005.
- [145] Charu C Aggarwal, Jiawei Han, Jianyong Wang, and Philip S Yu. A Framework for On-Demand Classification of Evolving Data Streams. *IEEE Transactions on Knowledge and Data Engineering*, 18 (5): 577–589, 2006.
- [146] Jing Gao, Wei Fan, and Jiawei Han. On Appropriate Assumptions to Mine Data Streams: Analysis and Practice. In *proceedings of the 7th IEEE International Conference on Data Mining (ICDM)*, pages 143–152. IEEE, 2007.
- [147] Albert Bifet, Gianmarco de Francisci Morales, Jesse Read, Geoff Holmes, and Bernhard Pfahringer. Efficient Online Evaluation of Big Data Stream Classifiers. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 59–68. ACM, 2015.
- [148] Lianbing Deng, Daming Li, Xiang Yao, David Cox, and Haoxiang Wang. Mobile network intrusion detection for IoT system based on transfer learning algorithm. *Cluster Computing*, pages 1–16, 2018.
- [149] DEFCON dataset. *DEF CON Hacking Conference*, 2018. URL <https://www.defcon.org/>, Accessed 21 May 2018.
- [150] CAIDA dataset. *Center for Applied Internet Data Analysis (CAIDA)*, 2018. URL <http://www.caida.org/>, Accessed 21 May 2018.
- [151] LBNL dataset. *LBNL/ICSI Enterprise Tracing Project*, 2005. URL <https://www.icir.org/enterprise-tracing/Overview.html>, Accessed 21 May 2018.
- [152] Monowar H Bhuyan, DK Bhattacharyya, and Jugal K Kalita. NADO: Network Anomaly Detection Using Outlier Approach. In *Proceedings of the 2011 International Conference on Communication, Computing & Security*, pages 531–536. ACM, 2011.

- [153] Monowar H Bhuyan, Dhruva K Bhattacharyya, and Jugal K Kalita. AOCD: An Adaptive Outlier Based Coordinated Scan Detection Approach. *International Journal of Network Security*, 14 (6): 339–351, 2012.
- [154] Prasanta Gogoi, Monowar H Bhuyan, DK Bhattacharyya, and Jugal K Kalita. Packet and Flow Based Network Intrusion Dataset. In *International Conference on Contemporary Computing*, pages 322–334. Springer, 2012.
- [155] Salvatore J Stolfo, Wei Fan, Wenke Lee, Andreas Prodromidis, and Philip K Chan. Cost-based Modeling for Fraud and Intrusion Detection: Results from the JAM Project. In *DARPA Information Survivability Conference and Exposition, 2000. DISCEX'00. Proceedings*, volume 2, pages 130–144. IEEE, 2000.
- [156] UCI KDD Archive. *KDD Cup 1999 Data*, 1999. URL <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>, Accessed 21 May 2018.
- [157] Mahbod Tavallaee, Ebrahim Bagheri, Wei Lu, and Ali A Ghorbani. A Detailed Analysis of the KDD CUP 99 Data Set. In *proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, pages 1–6. IEEE, 2009.
- [158] Jungsuk Song, Hiroki Takakura, and Yasuo Okabe. *Description of Kyoto University Benchmark Data*, 2006. URL http://www.takakura.com/Kyoto_data/BenchmarkData-Description-v5.pdf, Accessed 16 Jul 2018.
- [159] Iman Sharafaldin, A Habibi Lashkari, and Ali A Ghorbani. Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization. In *proceedings of 4th International Conference on Information Systems Security and Privacy (ICISSP)*, pages 108–116, 2018.
- [160] Iñigo Perona, Ibai Gurrutxaga, Olatz Arbelaitz, José I Martín, Javier Muguerza, and Jesús Ma Pérez. *gureKddcup database*, 2008. URL http://www.sc.ehu.es/acwaldap/gureKddcup/galdetegia_jaso.php, Accessed 17 Jul 2018.
- [161] Iñigo Perona, Ibai Gurrutxaga, Olatz Arbelaitz, José I Martín, Javier Muguerza, and Jesús Ma Pérez. Service-independent payload analysis to improve intrusion detection in network traffic . In *Proceedings of the 7th Australasian Data Mining Conference-Volume 87*, pages 171–178. Australian Computer Society, Inc., 2008.
- [162] Iñigo Perona, Olatz Arbelaitz Gallego, Ibai Gurrutxaga, José Ignacio Martín, Javier Francisco Muguerza Rivero, and Jesús María Pérez. *Generation of the database gurekddcup*, 2016. URL <http://hdl.handle.net/10810/20608>, Accessed 27 Jul 2018.
- [163] Anna Sperotto, Ramin Sadre, Frank Van Vliet, and Aiko Pras. A Labeled Data Set for Flow-Based Intrusion Detection. In *International Workshop on IP Operations and Management*, pages 39–50. Springer, 2009.
- [164] Nour Moustafa and Jill Slay. UNSW-NB15: A Comprehensive Data Set for Network Intrusion Detection Systems (UNSW-NB15 Network Data Set). In *Military Communications and Information Systems Conference (MilCIS), 2015*, pages 1–6. IEEE, 2015.
- [165] Nour Moustafa and Jill Slay. The evaluation of Network Anomaly Detection Systems: Statistical analysis of the UNSW-NB15 data set and the comparison with the KDD99 data set. *Information Security Journal: A Global Perspective*, 25 (1-3): 18–31, 2016.

- [166] Sebastian Abt and Harald Baier. Are We Missing Labels? A Study of the Availability of Ground-Truth in Network Security Research. In *Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS), 2014 Third International Workshop on*, pages 40–55. IEEE, 2014.
- [167] Xinxing Zhang, Hongri Liu, Bailing Wang, Junheng Huang, and Xixian Han. Generating Realistic Network Traffic and Interactive Application Workloads Using Container Technology. In *Communication Software and Networks (ICCSN), 2017 IEEE 9th International Conference on*, pages 1021–1025. IEEE, 2017.
- [168] Charles V Wright, Christopher Connelly, Timothy Braje, Jesse C Rabek, Lee M Rossey, and Robert K Cunningham. Generating Client Workloads and High-Fidelity Network Traffic for Controllable, Repeatable Experiments in Computer Security. In *International Workshop on Recent Advances in Intrusion Detection*, pages 218–237. Springer, 2010.
- [169] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning*, volume 112. Springer, 2013.
- [170] Mairead L Bermingham, Ricardo Pong-Wong, Athina Spiliopoulou, Caroline Hayward, Igor Rudan, Harry Campbell, Alan F Wright, James F Wilson, Felix Agakov, Pau Navarro, and C Haley. Application of high-dimensional feature selection: evaluation for genomic prediction in man. *Scientific Reports*, 5: 1–12, 2015.
- [171] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction (Second Edition)*, volume 1. Springer series in statistics New York, 2011.
- [172] Jason Brownlee. *An Introduction to Feature Selection*, 2014. URL <https://machinelearningmastery.com/an-introduction-to-feature-selection/>, Accessed 16 Jul 2018.
- [173] Jiliang Tang, Salem Alelyani, and Huan Liu. Feature Selection for Classification: A Review. *Data Classification: Algorithms and Applications*, pages 1–33, 2014.
- [174] Salem Alelyani, Jiliang Tang, and Huan Liu. Feature Selection for Clustering: A Review. *Data Clustering: Algorithms and Applications*, 29: 110–121, 2013.
- [175] Ron Kohavi and George H John. Wrappers for Feature Subset Selection. *Artificial Intelligence*, 97 (1-2): 273–324, 1997.
- [176] Volker Roth and Tilman Lange. Feature Selection in Clustering Problems. In *Advances in Neural Information Processing Systems*, pages 473–480, 2004.
- [177] Stuart J Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (3rd Edition)*. Malaysia; Pearson Education Limited., 2016. ISBN 0136042597.
- [178] Judea Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, 1984.
- [179] Steven S. Skiena. *The Algorithm Design Manual - Second Edition*. Springer Science & Business Media, 2008.
- [180] Alejandro Figueroa. Exploring effective features for recognizing the user intent behind web queries. *Computers in Industry*, 68: 162–169, 2015.
- [181] Alejandro Figueroa and Günter Neumann. Category-Specific Models for Ranking Effective Paraphrases in Community Question Answering. *Expert Systems with Applications*, 41 (10): 4730–4742, 2014.

- [182] Alejandro Figueroa and Günter Neumann. Learning to Rank Effective Paraphrases from Query Logs for Community Question Answering. In *proceedings of the 27th AAAI Conference on Artificial Intelligence*, volume 13, pages 1099–1105, 2013.
- [183] Manoranjan Dash and Huan Liu. Feature Selection for Clustering. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 110–121. Springer, 2000.
- [184] Huan Liu and Rudy Setiono. A Probabilistic Approach to Feature Selection-A Filter Solution. In *proceedings of the 13th International Conference on International Conference on Machine Learning*, volume 96, pages 319–327. Citeseer, 1996.
- [185] Kenji Kira and Larry A Rendell. The Feature Selection Problem: Traditional Methods and a New Algorithm. In *proceedings of the 10th National Conference on Artificial Intelligence*, volume 2, pages 129–134, 1992.
- [186] Kenji Kira and Larry A Rendell. A Practical Approach to Feature Selection. In *Machine Learning Proceedings 1992*, pages 249–256. Elsevier, 1992.
- [187] Ryan J Urbanowicz, Melissa Meeker, William LaCava, Randal S Olson, and Jason H Moore. Relief-Based Feature Selection: Introduction and Review. *Journal of Biomedical Informatics*, pages 1532–0464, 2018.
- [188] Isabelle Guyon and André Elisseeff. An Introduction to Variable and Feature Selection. *Journal of machine learning research*, 3 (Mar): 1157–1182, 2003.
- [189] Thomas M Cover and Joy A Thomas. *Elements of Information Theory (Second Edition)*. John Wiley & Sons, Inc., 1991.
- [190] Hanchuan Peng, Fuhui Long, and Chris Ding. Feature Selection Based on Mutual Information Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy. *IEEE Transactions on pattern analysis and machine intelligence*, 27 (8): 1226–1238, 2005.
- [191] Karl Pearson. Contributions to the Mathematical Theory of Evolution. In *Proceedings of the Royal Society of London*, volume 54, pages 329–333. JSTOR, 1893.
- [192] Karl Pearson. Contributions to the Mathematical Theory of Evolution. II. Skew Variation in Homogeneous Material. *Philosophical transactions of the Royal Society of London*, 186 (Part I): 343–424, 1895.
- [193] Karl Pearson, Alice Lee, Ernest Warren, Agnes Fry, and Cicely D Fawcett. Mathematical contributions to the theory of evolution. IX.—On the principle of homotyposis and its relation to heredity, to the variability of the individual, and to that of the race. Part I.—homotyposis in the vegetable kingdom. *Proceedings of the Royal Society of London*, 68 (442-450): 1–5, 1901.
- [194] Sanmay Das. Filters, Wrappers and a Boosting-Based Hybrid for Feature Selection. In *proceedings of the 18th International Conference on Machine Learning*, volume 1, pages 74–81, 2001.
- [195] Andrew Y Ng. *On Feature Selection: Learning with Exponentially many Irrelevant Features as Training Examples*. PhD thesis, Massachusetts Institute of Technology, 1998.
- [196] Robert Tibshirani. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

- [197] Andrew Y Ng. Feature selection, L1 vs. L2 regularization, and rotational invariance. In *Proceedings of the 21st International Conference on Machine Learning*, pages 78–94. ACM, 2004.
- [198] Hui Zou and Trevor Hastie. Regularization and variable selection via the elastic net. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 67 (2): 301–320, 2005.
- [199] Arash Habibi Lashkari, Gerard Draper-Gil, Mohammad Saiful Islam Mamun, and Ali A Ghorbani. Characterization of Tor Traffic using Time based Features. In *proceedings of the 3rd International Conference on Information Systems Security and Privacy (ICISSP)*.
- [200] M. A. Hall. *Correlation-Based Feature Subset Selection for Machine Learning*. PhD thesis, University of Waikato, Hamilton, New Zealand, 1998.
- [201] Shadi Aljawarneh, Monther Aldwairi, and Muneer Bani Yassein. Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model. *Journal of Computational Science*, pages 152–160, 2017.
- [202] Mohammed A Ambusaidi, Xiangjian He, Priyadarsi Nanda, and Zhiyuan Tan. Building an Intrusion Detection System Using a Filter-Based Feature Selection Algorithm. *IEEE Transactions on Computers*, 65 (10): 2986–2998, 2016.
- [203] Yu Gu, Andrew McCallum, and Don Towsley. Detecting Anomalies in Network Traffic Using Maximum Entropy Estimation. In *proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement*, pages 32–32. USENIX Association, 2005.
- [204] Ayesha Binte Ashfaq, Maria Joseph Robert, Asma Mumtaz, Muhammad Qasim Ali, Ali Sajjad, and Syed Ali Khayam. A Comparative Evaluation of Anomaly Detectors under Portscan Attacks. In *International Workshop on Recent Advances in Intrusion Detection*, pages 351–371. Springer, 2008.
- [205] Christophe Ambroise and Geoffrey J. McLachlan. Selection bias in gene extraction on the basis of microarray gene-expression data. *Proceedings of the National Academy of Sciences*, 99 (10): 6562–6566, 2002. URL <http://www.pnas.org/content/99/10/6562>.
- [206] Nathalie Japkowicz. The Class Imbalance Problem: Significance and Strategies. In *Proceedings of the 2000 International Conference on Artificial Intelligence ICAI*, pages 111–117, 2000.
- [207] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. SMOTE: Synthetic Minority Over-sampling Technique. *Journal of artificial intelligence research*, 16: 321–357, 2002.
- [208] Ivan Tomek. Two Modifications of CNN. *IEEE Trans. Systems, Man and Cybernetics*, 6: 769–772, 1976.
- [209] Show-Jane Yen and Yue-Shi Lee. Cluster-based under-sampling approaches for imbalanced data distributions. *Expert Systems with Applications*, 36 (3): 5718–5727, 2009.
- [210] Chumphol Bunkhumpornpat and Krung Sinapiromsaran. DBMUTE: density-based majority under-sampling technique. *Knowledge and Information Systems*, 50 (3): 827–850, 2017.

- [211] Salvador García and Francisco Herrera. Evolutionary Undersampling for Classification with Imbalanced Datasets: Proposals and Taxonomy. *Evolutionary computation*, 17 (3): 275–306, 2009.
- [212] Jihyun Ha and Jong-Seok Lee. A New Under-Sampling Method Using Genetic Algorithm for Imbalanced Data Classification. In *Proceedings of the 10th International Conference on Ubiquitous Information Management and Communication*, pages 95:1–95:6. ACM, 2016.
- [213] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning. In *International Conference on Intelligent Computing*, pages 878–887. Springer, 2005.
- [214] Chumphol Bunkhumpornpat, Krung Sinapiromsaran, and Chidchanok Lursinsap. Safe-Level-Smote: Safe-Level-Synthetic Minority Over-Sampling Technique for Handling the Class Imbalanced Problem. In *Pacific-Asia conference on knowledge discovery and data mining*, pages 475–482. Springer, 2009.
- [215] Lida Abdi and Sattar Hashemi. To Combat Multi-Class Imbalanced Problems by Means of Over-Sampling Techniques. *IEEE transactions on Knowledge and Data Engineering*, 28 (1): 238–251, 2016.
- [216] Chumphol Bunkhumpornpat, Krung Sinapiromsaran, and Chidchanok Lursinsap. DBSMOTE: Density-Based Synthetic Minority Over-Sampling TEchnique. *Applied Intelligence*, 36 (3): 664–684, 2012.
- [217] Haibo He, Yang Bai, Eduardo A Garcia, and Shutao Li. ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning. In *Neural Networks, 2008. IJCNN 2008.(IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, pages 1322–1328. IEEE, 2008.
- [218] Bartosz Krawczyk and Michał Woźniak. Diversity Measures for One-Class Classifier Ensembles. *Neurocomputing*, 126: 36–44, 2014.
- [219] Bartosz Krawczyk, Michał Woźniak, and Francisco Herrera. Weighted One-Class Classification for Different Types of Minority Class Examples in Imbalanced Data. In *Computational Intelligence and Data Mining (CIDM), 2014 IEEE Symposium on*, pages 337–344. IEEE, 2014.
- [220] Bartosz Krawczyk, Łukasz Jeleń, Adam Krzyżak, and Thomas Fevens. One-Class Classification Decomposition for Imbalanced Classification of Breast Cancer Malignancy Data. In *International Conference on Artificial Intelligence and Soft Computing*, pages 539–550. Springer, 2014.
- [221] Hwanjo Yu. Single-Class Classification with Mapping Convergence. *Machine Learning*, 61 (1-3): 49–69, 2005.
- [222] Shehroz S Khan and Michael G Madden. A Survey of Recent Trends in One Class Classification. In *Irish Conference on Artificial Intelligence and Cognitive Science*, pages 188–197. Springer, 2009.
- [223] Guillaume Lemaître, Fernando Nogueira, and Christos K Aridas. Imbalanced-Learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. *Journal of Machine Learning Research*, 18 (17): 1–5, 2017.
- [224] Nathalie Japkowicz and Shaju Stephen. The Class Imbalance Problem: A Systematic Study. *Intelligent data analysis*, 6 (5): 429–449, 2002.

- [225] Alan H Fielding and John F Bell. A Review of Methods for the Assessment of Prediction Errors in Conservation Presence/Absence Models. *Environmental conservation*, 24 (1): 38–49, 1997.
- [226] David L Verbyla and John A Litvaitis. Resampling Methods for Evaluating Classification Accuracy of Wildlife Habitat Models. *Environmental Management*, 13 (6): 783–787, 1989.
- [227] David Stockwell. *Machine Learning and the Problem of Prediction and Explanation in Ecological Modelling*. PhD thesis, Ecosystem Dynamics Group, Research School of Biological Sciences, The Australian National University, 1992.
- [228] Stephanie Sundberg. *Prospective Study: Definition, Examples*, 2018. URL <http://www.statisticshowto.com/prospective-study/>, Accessed 16 Jul 2018.
- [229] Tom Fawcett. An Introduction to ROC Analysis. *Pattern recognition letters*, 27 (8): 861–874, 2006.
- [230] David Hand and Peter Christen. A Note on Using the F-Measure for Evaluating Record Linkage Algorithms. *Statistics and Computing*, pages 1–9, 2017.
- [231] Miroslav Kubat and Stan Matwin. Addressing the Curse of Imbalanced Training Sets: One-Sided Selection. In *proceedings of the 14th International Conference on Machine Learning (ICML97)*, volume 97, pages 179–186. Nashville, USA, 1997.
- [232] Ludmila I. Kuncheva, Álvar Arnaiz-González, José-Francisco Díez-Pastor, and Iain A. D. Gunn. Instance Selection Improves Geometric Mean Accuracy: A Study on Imbalanced Data Classification. *CoRR*, abs/1804.07155: 1–11, 2018. URL <http://arxiv.org/abs/1804.07155>, Accessed 16 Jul 2018.
- [233] C Madhusudhana Rao and MM Naidu. A Model for Generating Synthetic Network Flows and Accuracy Index for Evaluation of Anomaly Network Intrusion Detection Systems. *Indian Journal of Science and Technology*, 10 (14): 1–16, 2017.
- [234] C Madhusudhana Rao and MM Naidu. Acceptance Sampling for Network Intrusion Detection. *Journal of Theoretical & Applied Information Technology*, 95 (24): 6707–6718, 2017.
- [235] Santiago Beguería. Validation and Evaluation of Predictive Models in Hazard Assessment and Risk Management. *Natural Hazards*, 37 (3): 315–329, Mar 2006. URL <https://doi.org/10.1007/s11069-005-5182-6>, Accessed 16 Jul 2018.
- [236] Kevin Markham. *Simple Guide to Confusion Matrix Terminology*, 2014. URL <http://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/>, Accessed 16 Jul 2018.
- [237] Mark H Zweig and Gregory Campbell. Receiver-Operating Characteristic (ROC) Plots: A Fundamental Evaluation Tool in Clinical Medicine. *Clinical chemistry*, 39 (4): 561–577, 1993.
- [238] Elizabeth R DeLong, David M DeLong, and Daniel L Clarke-Pearson. Comparing the Areas Under Two or More Correlated Receiver Operating Characteristic Curves: A Nonparametric Approach. *Biometrics*, pages 837–845, 1988.
- [239] James A Hanley and Barbara J McNeil. The Meaning and Use of the Area under a Receiver Operating Characteristic (ROC) Curve. *Radiology*, 143 (1): 29–36, 1982.

- [240] Manish Barnwal. *ROC and AUC - the Three Lettered Acronyms*, 2016. URL http://manishbarnwal.com/blog/2016/09/26/three_letter_acronym_roc_and_auc/, Accessed 16 Jul 2018.
- [241] Paolo Verme and Chiara Gigliarano. Optimal targeting under budget constraints in a humanitarian context. *World Development*, 2018. URL <https://doi.org/10.1016/j.worlddev.2017.12.012>, Accessed 16 Jul 2018.
- [242] Blaise Hanczar, Jianping Hua, Chao Sima, John Weinstein, Michael Bittner, and Edward R Dougherty. Small-Sample Precision of ROC-Related Estimates. *Bioinformatics*, 26 (6): 822–830, 2010.
- [243] Jorge M Lobo, Alberto Jiménez-Valverde, and Raimundo Real. AUC: a misleading measure of the performance of predictive distribution models. *Global ecology and Biogeography*, 17 (2): 145–151, 2008.
- [244] David J Hand. Measuring classifier performance: a coherent alternative to the area under the ROC curve. *Machine learning*, 77 (1): 103–123, 2009.
- [245] Umberto Lucchetti. *AML Rule Tuning: Applying Statistical and Risk-Based Approach to Achieve Higher Alert Efficiency*, 2015. URL <http://www.acams.org/wp-content/uploads/2015/08/AML-Rule-Tuning-Applying-Statistical-Risk-Based-Approach-to-Achieve-Higher-Alert-Efficiency-U-Lucchetti.pdf>, Accessed 21 May 2018.
- [246] J. J. Chen, C.-A. Tsai, H. Moon, H. Ahn, J. J. Young, and C.-H. Chen. Decision Threshold Adjustment in Class Prediction. *SAR and QSAR in Environmental Research*, 17 (3): 337–352, 2006. URL <https://doi.org/10.1080/10659360600787700>.
- [247] Elizabeth A Freeman and Gretchen G Moisen. A comparison of the performance of threshold criteria for binary classification in terms of predicted prevalence and kappa. *Ecological Modelling*, 217 (1-2): 48–58, 2008.
- [248] Yiming Yang. A Study of Thresholding Strategies for Text Categorization. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 137–145. ACM, 2001. URL <http://doi.acm.org/10.1145/383952.383975>.
- [249] Anukool Lakhina, Mark Crovella, and Christophe Diot. Diagnosing Network-Wide Traffic Anomalies. In *ACM SIGCOMM Computer Communication Review*, volume 34, pages 219–230. ACM, 2004.
- [250] Rong-En Fan and Chih-Jen Lin. A Study on Threshold Selection for Multi-Label Classification. *Department of Computer Science, National Taiwan University*, pages 1–23, 2007.
- [251] Ignazio Pillai, Giorgio Fumera, and Fabio Roli. Threshold Optimisation for Multi-Label Classifiers. *Pattern Recognition*, 46 (7): 2055–2065, 2013.
- [252] Oluwasanmi O Koyejo, Nagarajan Natarajan, Pradeep K Ravikumar, and Inderjit S Dhillon. Consistent Binary Classification with Generalized Performance Metrics. In *Advances in Neural Information Processing Systems*, pages 2744–2752, 2014.
- [253] Bowei Yan, Oluwasanmi Koyejo, Kai Zhong, and Pradeep Ravikumar. Binary Classification with Karmic, Threshold-Quasi-Concave Metrics. *arXiv preprint arXiv:1806.00640*, 2018.

- [254] Eleazar Eskin, Matthew Miller, Zhi-Da Zhong, George Yi, Wei-Ang Lee, and Salvatore Stolfo. Adaptive Model Generation for Intrusion Detection Systems. In *Proceedings of the ACMCCS Workshop on Intrusion Detection and Prevention, Athens, Greece*, pages 1–14, 2000.
- [255] Andrew Honig, Andrew Howard, Eleazar Eskin, and Salvatore Stolfo. Adaptive Model Generation: An Architecture for Deployment of Data Mining-Based Intrusion Detection Systems. In *proceedings of the Applications of Data Mining in Computer Security*, pages 153–194. Kluwer Academic Publishers, 2002.
- [256] Mahmood Hossain and Susan M Bridges. A Framework for an Adaptive Intrusion Detection System with Data Mining. In *proceedings of the 13th Annual Canadian Information Technology Security Symposium*, pages 1–8, 2001.
- [257] Mahmood Hossain, Susan M Bridges, and Rayford B Vaughn. Adaptive Intrusion Detection with Data Mining. In *Systems, Man and Cybernetics, 2003. IEEE International Conference on*, volume 4, pages 3097–3103. IEEE, 2003.
- [258] Jaeyeon Jung, V. Paxson, A. W. Berger, and H. Balakrishnan. Fast Portscan Detection Using Sequential Hypothesis Testing. In *IEEE Symposium on Security and Privacy, 2004. Proceedings. 2004*, pages 211–225, May 2004. doi: 10.1109/SECPRI.2004.1301325.
- [259] Muhammad Qasim Ali, Ehab Al-Shaer, Hassan Khan, and Syed Ali Khayam. Automated Anomaly Detector Adaptation Using Adaptive Threshold Tuning. *ACM Transactions on Information and System Security (TISSEC)*, 15 (4): 17, 2013.
- [260] Tsuyoshi Idé and Hisashi Kashima. Eigenspace-Based Anomaly Detection in Computer Systems. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '04*, pages 440–449, New York, NY, USA, 2004. ACM. ISBN 1-58113-888-1. doi: 10.1145/1014052.1014102. URL <http://doi.acm.org/10.1145/1014052.1014102>.
- [261] Z. Yu, J. J. P. Tsai, and T. Weigert. An Automatically Tuning Intrusion Detection System. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 37 (2): 373–384, April 2007. ISSN 1083-4419. doi: 10.1109/TSMCB.2006.885306.
- [262] Zhenwei Yu, Jeffrey J. P. Tsai, and Thomas Weigert. An Adaptive Automatically Tuning Intrusion Detection System. *ACM Trans. Auton. Adapt. Syst.*, 3 (3): 10:1–10:25, August 2008. ISSN 1556-4665. doi: 10.1145/1380422.1380425. URL <http://doi.acm.org/10.1145/1380422.1380425>.
- [263] Stephanie Forrest, Steven A Hofmeyr, Anil Somayaji, and Thomas A Longstaff. A Sense of Self for Unix Processes. In *Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on*, pages 120–128. IEEE, 1996.
- [264] Dae-Ki Kang, Doug Fuller, and Vasant Honavar. Learning Classifiers for Misuse and Anomaly Detection Using a Bag of System Calls Representation. In *Information Assurance Workshop, 2005. IAW'05. Proceedings from the Sixth Annual IEEE SMC*, pages 118–125. IEEE, 2005.
- [265] Debin Gao, Michael K Reiter, and Dawn Song. Behavioral Distance for Intrusion Detection. In *International Workshop on Recent Advances in Intrusion Detection*, pages 63–81. Springer, 2005.

- [266] Andrew H Sung, Jianyun Xu, Patrick Chavez, and Srinivas Mukkamala. Static Analyzer of Vicious Executables (Save). In *Computer Security Applications Conference, 2004. 20th Annual*, pages 326–334. IEEE, 2004.
- [267] Hui-Hao Chou and Sheng-De Wang. An Adaptive Network Intrusion Detection Approach for the Cloud Environment. In *Security Technology (ICCST), 2015 International Carnahan Conference on*, pages 1–6. IEEE, 2015.
- [268] John Mark Agosta, Carlos Diuk-Wasser, Jaideep Chandrashekar, and Carl Livadas. An Adaptive Anomaly Detector for Worm Detection. In *Proceedings of the 2Nd USENIX Workshop on Tackling Computer Systems Problems with Machine Learning Techniques, SYSMML'07*, pages 3:1–3:6, Berkeley, CA, USA, 2007. USENIX Association. URL <http://dl.acm.org/citation.cfm?id=1361442.1361445>.
- [269] Guofei Gu, Prahlad Fogla, David Dagon, Wenke Lee, and Boris Skoric. Towards an Information-Theoretic Framework for Analyzing Intrusion Detection Systems. In *European Symposium on Research in Computer Security*, pages 527–546. Springer, 2006.
- [270] Chris Strasburg, Samik Basu, and Johnny S Wong. S-MAIDS: A Semantic Model for Automated Tuning, Correlation, and Response Selection in Intrusion Detection Systems. In *Computer Software and Applications Conference (COMPSAC), 2013 IEEE 37th Annual*, pages 319–328. IEEE, 2013.
- [271] V. Jyothisna and V. V. Rama Prasad. Assessing Degree of Intrusion Scope (DIS): A Statistical Strategy for Anomaly Based Intrusion Detection. *CSI Transactions on ICT*, pages 1–29, Mar 2018. doi: 10.1007/s40012-018-0188-x. URL <https://doi.org/10.1007/s40012-018-0188-x>.
- [272] Albert Bifet, Geoff Holmes, Bernhard Pfahringer, Richard Kirkby, and Ricard Gavaldà. New Ensemble Methods for Evolving Data Streams. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 139–148. ACM, 2009.
- [273] Mohammad Masud, Jing Gao, Latifur Khan, Jiawei Han, and Bhavani M Thuraisingham. Classification and Novel Class Detection in Concept-Drifting Data Streams under Time Constraints. *IEEE Transactions on Knowledge and Data Engineering*, 23 (6): 859–874, 2011.
- [274] Dewan Md Farid, Li Zhang, Alamgir Hossain, Chowdhury Mofizur Rahman, Rebecca Strachan, Graham Sexton, and Keshav Dahal. An Adaptive Ensemble Classifier for Mining Concept Drifting Data Streams. *Expert Systems with Applications*, 40 (15): 5895–5906, 2013.
- [275] Mohammad M Masud, Qing Chen, Latifur Khan, Charu Aggarwal, Jing Gao, Jiawei Han, and Bhavani Thuraisingham. Addressing Concept-Evolution in Concept-Drifting Data Streams. In *Data Mining (ICDM), 2010 IEEE 10th International Conference on*, pages 929–934. IEEE, 2010.
- [276] Mohammad M Masud, Qing Chen, Latifur Khan, Charu C Aggarwal, Jing Gao, Jiawei Han, Ashok Srivastava, and Nikunj C Oza. Classification and Adaptive Novel Class Detection of Feature-Evolving Data Streams. *IEEE Transactions on Knowledge and Data Engineering*, 25 (7): 1484–1497, 2013.
- [277] Gabriela F Cretu-Ciocarlie, Angelos Stavrou, Michael E Locasto, and Salvatore J Stolfo. Adaptive Anomaly Detection via Self-Calibration and Dynamic Updating. In

- International Workshop on Recent Advances in Intrusion Detection*, pages 41–60. Springer, 2009.
- [278] Shixi Chen, Haixun Wang, Shuigeng Zhou, and S Yu Philip. Stop Chasing Trends: Discovering High Order Models in Evolving Data. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 923–932. IEEE, 2008.
- [279] Heitor M Gomes, Albert Bifet, Jesse Read, Jean Paul Barddal, Fabrício Enembreck, Bernhard Pfharinger, Geoff Holmes, and Talel Abdessalem. Adaptive Random Forests for Evolving Data Stream Classification. *Machine Learning*, 106 (9-10): 1469–1495, 2017.
- [280] Wojciech Kotłowski and Krzysztof Dembczyński. Surrogate Regret Bounds for Generalized Classification Performance Metrics. *Machine Learning*, 106 (4): 549–572, Apr 2017. ISSN 1573-0565. doi: 10.1007/s10994-016-5591-7. URL <https://doi.org/10.1007/s10994-016-5591-7>.
- [281] Tomasz Bujlow, Tahir Riaz, and Jens Myrup Pedersen. A method for classification of network traffic based on C5.0 Machine Learning Algorithm. In *Computing, Networking and Communications (ICNC), 2012 International Conference on*, pages 237–241. IEEE, 2012.
- [282] aporras. *What Is the Difference Between Bagging and Boosting?*, 2016. URL <https://quantdare.com/what-is-the-difference-between-bagging-and-boosting/>, Accessed 21 May 2018.
- [283] Rulequest Research. *Is See5/C5.0 Better Than C4.5?*, 2017. URL <http://rulequest.com/see5-comparison.html>, Accessed 30 Jun 2018.
- [284] Raghav Aggiwal. *Introduction to Random Forest*, Feb 2017. URL <https://dimensionless.in/tag/random-forest/>, Accessed 30 Jun 2018.
- [285] L Breiman, J Friedman, R Olshen, and C Stone. *Classification and Regression Trees*. Monterey, CA: Wadsworth & Brooks/Cole Advanced Books & Software, 1984.
- [286] Saul B Gelfand, CS Ravishankar, and Edward J Delp. An Iterative Growing and Pruning Algorithm for Classification Tree Design. In *IEEE International Conference Proceedings on Systems, Man and Cybernetics.*, pages 818–823. IEEE, 1989.
- [287] J. Ross Quinlan. Simplifying Decision Trees. *International Journal of Man-Machine Studies*, 27 (3): 221–234, 1987. URL [https://doi.org/10.1016/S0020-7373\(87\)80053-6](https://doi.org/10.1016/S0020-7373(87)80053-6), Accessed 21 May 2018.
- [288] Zdravko Botev and Ad Ridder. Variance Reduction. *Wiley StatsRef: Statistics Reference Online*, pages 1–6, 2017.
- [289] Tom Mitchell. *Machine Learning*. The Mc-Graw-Hill Companies, Inc., 1997. URL <https://www.cs.ubbcluj.ro/~gabis/ml/ml-books/McGrawHill%20-%20Machine%20Learning%20-Tom%20Mitchell.pdf>.
- [290] J. Ross Quinlan. Induction of Decision Trees. *Machine learning*, 1 (1): 81–106, 1986.
- [291] J Ross Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kauffmann Publishers, Inc., 1993.
- [292] Rulequest Research. *C5.0: An Informal Tutorial*, 2017. URL <https://www.rulequest.com/see5-unix.html>, Accessed 21 May 2018.

- [293] Bradley Efron and Robert J Tibshirani. *An Introduction to the Bootstrap*. CRC press, 1994. URL <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.473.2742&rep=rep1&type=pdf>, Accessed 21 May 2018.
- [294] Leo Breiman. Bagging Predictors. *Machine learning*, 24 (2): 123–140, 1996.
- [295] Chunyang Li. Probability Estimation in Random Forests. Master’s thesis, Mathematics and Statistics, Utah State University, 2013. URL <https://digitalcommons.usu.edu/cgi/viewcontent.cgi?article=1304&context=gradreports>, Accessed 21 May 2018.
- [296] James D Malley, Jochen Kruppa, Abhijit Dasgupta, Karen G Malley, and Andreas Ziegler. Probability Machines: Consistent Probability Estimation Using Nonparametric Learning Machines. *Methods of Information in Medicine*, 51 (1): 74–81, 2012.
- [297] Marvin N. Wright, Stefan Wager, and Philipp Probst. *Ranger: A Fast Implementation of Random Forests*, 2017. Version 0.8.0, URL <https://CRAN.R-project.org/package=ranger>, Accessed 21 May 2018.
- [298] Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann, 2016.
- [299] Paulo Angelo Alves Resende and André Costa Drummond. A Survey of Random Forest Based Methods for Intrusion Detection Systems. *ACM Computing Surveys (CSUR)*, 51 (3): 48:1–48:36, 2018.
- [300] Taghi M Khoshgoftaar, Moiz Golawala, and Jason Van Hulse. An Empirical Study of Learning from Imbalanced Data Using Random Forest. In *Tools with Artificial Intelligence, 2007. ICTAI 2007. 19th IEEE international conference on*, volume 2, pages 310–317. IEEE, 2007.
- [301] Shih-Wei Lin, Kuo-Ching Ying, Chou-Yuan Lee, and Zne-Jung Lee. An intelligent algorithm with feature selection and decision rules applied to anomaly intrusion detection . *Applied Soft Computing*, 12 (10): 3285–3290, 2012.
- [302] Corinna Cortes and Vladimir Vapnik. Support-Vector Networks. *Machine learning*, 20 (3): 273–297, 1995.
- [303] Vladimir Vapnik. *Statistical Learning Theory*. John Wiley and Sons, Inc., New York, 1998.
- [304] Vahid Golmah. An Efficient Hybrid Intrusion Detection System Based on C5. 0 And SVM. *International Journal of Database Theory and Application*, 7 (2): 59–70, 2014.
- [305] Alexander Statnikov, Douglas Hardin, Isabelle Guyon, and Constantin F. Aliferis. A Gentle Introduction to Support Vector Machines in Biomedicine. In *AMIA Annual Symposium Proceedings*. American Medical Informatics Association, New York University - School of Medicin, 2009. URL <https://med.nyu.edu/chibi/sites/default/files/chibi/Final.pdf>, Accessed 30 Jun 2018.
- [306] Tom Kelsey. *Lecture Notes - ID5059: Knowledge Discovery and Data Mining, Lecture 17 - Support Vector Machines (SVMs)*, 2017. University of St Andrews.
- [307] Dikran Marsupial. *SVM, Overfitting, Curse of Dimensionality*, Aug 2012. URL <https://stats.stackexchange.com/questions/35276/svm-overfitting-curse-of-dimensionality>, Accessed 30 Jun 2018.

- [308] Tom Kelsey. *Lecture Notes - ID5059: Knowledge Discovery and Data Mining, Lecture 18 - Support Vector Machines (SVMs) (2)*, 2017. University of St Andrews.
- [309] Yoav Goldberg and Michael Elhadad. splitSVM: Fast, Space-Efficient, non-Heuristic, Polynomial Kernel Computation for NLP Applications. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, pages 237–240. Association for Computational Linguistics, 2008.
- [310] Bernhard Schölkopf, Koji Tsuda, and Jean-Philippe Vert. *Kernel Methods in Computational Biology, Chapter 2: A Primer on Kernel Methods*. MIT press, 2004.
- [311] John Platt. Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. *Advances in large margin classifiers*, 10 (3): 61–74, 1999.
- [312] Michael Kemmler. *Question - Can We Assign Probability to SVM Results Instead of a Binary Output?*, 2013. URL https://www.researchgate.net/post/Can_we_assign_probability_to_SVM_results_instead_of_a_binary_output2, Accessed 30 Jun 2018.
- [313] Hsuan-Tien Lin, Chih-Jen Lin, and Ruby C Weng. A note on Platt’s probabilistic outputs for support vector machines. *Machine learning*, 68 (3): 267–276, 2007.
- [314] Grzegorz Gwardys. *Why Is Kernelized SVM Much Slower Than Linear SVM?*, Nov 2011. URL <https://www.quora.com/Why-is-kernelized-SVM-much-slower-than-linear-SVM>, Accessed 30 Jun 2018.
- [315] Christopher JC Burges. A Tutorial on Support Vector Machines for Pattern Recognition. *Data mining and knowledge discovery*, 2 (2): 121–167, 1998.
- [316] Chih-Jen Lin. *Chih-Jen Lin’s Home Page*. URL <https://www.csie.ntu.edu.tw/~cjlin/>, Accessed 26 Feb 2018.
- [317] W Nick Street and YongSeog Kim. A Streaming Ensemble Algorithm (SEA) for Large-Scale Classification. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 377–382. ACM, 2001.
- [318] Rakesh Agrawal, Tomasz Imielinski, and Arun Swami. Database Mining: A Performance Perspective. *IEEE Transactions on Knowledge and Data Engineering*, 5 (6): 914–925, 1993.
- [319] Pavel Laskov, Patrick Düssel, Christin Schäfer, and Konrad Rieck. Learning Intrusion Detection: Supervised or Unsupervised? In Fabio Roli and Sergio Vitulano, editors, *Image Analysis and Processing – ICIAP 2005*, pages 50–57, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.
- [320] Max Kuhn, Steve Weston, Nathan Coulter, Mark Culp, and C code for C5.0 by R. Quinlan. *C50: C5.0 Decision Trees and Rule-Based Models*, 2015. Version 0.1.0-24, URL <https://CRAN.R-project.org/package=C50>, Accessed 21 May 2018.
- [321] R Development Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2008. ISBN 3-900051-07-0, URL <http://www.R-project.org>, Accessed 21 May 2018.
- [322] Marvin N. Wright and Andreas Ziegler. Ranger: A Fast Implementation of Random Forests for High Dimensional Data in C++ and R. *Journal of Statistical Software*, 77 (1): 1–17, 2017. doi: 10.18637/jss.v077.i01.

- [323] Thibault Helleputte, Pierre Gramme, and Jerome Paul. LiblineaR: Linear Predictive Models Based on the LIBLINEAR C/C++ Library. *R package version 2.10-8*, pages 1–11, 2017. URL <https://CRAN.R-project.org/package=LiblineaR>, Accessed 21 May 2018.
- [324] Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A Library for Large Linear Classification. *Journal of machine learning research*, 9 (Aug): 1871–1874, 2008.
- [325] Susan Garavaglia and Asha Sharma. A Smart Guide to Dummy Variables: Four Applications and a Macro. In *proceedings of the Northeast SAS Users Group Conference*, pages 46–55, 1998.
- [326] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A Library for Support Vector Machines. *ACM transactions on intelligent systems and technology (TIST)*, 2 (3): 27:1–27:27, 2011.
- [327] Seymour Geisser. *Predictive Inference*. New York, NY: Chapman and Hall, 1993.
- [328] Ron Kohavi. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. In *Ijcai*, volume 14, pages 1137–1145. Montreal, Canada, 1995.
- [329] Pierre A Devijver and Josef Kittler. *Pattern Recognition: A Statistical Approach*. London, GB: Prentice hall, 1982.
- [330] Giovanni Seni and John F Elder. Ensemble Methods in Data Mining: Improving Accuracy Through Combining Predictions. *Synthesis Lectures on Data Mining and Knowledge Discovery*, 2 (1): 1–126, 2010.
- [331] Prashant Gupta. *Cross-Validation in Machine Learning*, 2017. URL <https://towardsdatascience.com/cross-validation-in-machine-learning-72924a69872f>, Accessed 16 Jul 2018.
- [332] Samuel Sanford Shapiro and Martin B Wilk. An analysis of variance test for normality (complete samples). *Biometrika*, 52 (3/4): 591–611, 1965.
- [333] Theodore W Anderson and Donald A Darling. Asymptotic Theory of Certain "Goodness of Fit" Criteria Based on Stochastic Processes. *The Annals of Mathematical Statistics*, 23 (2): 193–212, 1952.
- [334] Theodore W Anderson and Donald A Darling. A Test of Goodness of Fit. *Journal of the American Statistical Association*, 49 (268): 765–769, 1954.
- [335] Milton Friedman. The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance. *Journal of the American Statistical Association*, 32 (200): 675–701, 1937.
- [336] Milton Friedman. A Comparison of Alternative Tests of Significance for the Problem of m Rankings. *The Annals of Mathematical Statistics*, 11 (1): 86–92, 1940.
- [337] William Jay Conover. *Practical Nonparametric Statistics (Third Edition)*. Wiley New York, 1999. ISBN 0471160687.
- [338] Myles Hollander, Douglas A. Wolfe, and Eric Chicken. *Nonparametric Statistical Methods (Third Edition)*. John Wiley & Sons, 2013. ISBN 9781118553299. URL <https://www.wiley.com/en-us/Nonparametric+Statistical+Methods%2C+3rd+Edition-p-9780470387375>, Accessed 16 Jul 2018.

- [339] Ms Snehlata Dongre and Latesh Malik. Algorithm to Handle Concept Drifting in Data Stream Mining. *International Journal of Computer Science and Network (IJCSN)*, 2 (1): 107–111, 2013.
- [340] Ryan Elwell and Robi Polikar. Incremental Learning of Concept Drift in Nonstationary Environments. *IEEE Transactions on Neural Networks*, 22 (10): 1517–1531, 2011.
- [341] Charu C Aggarwal. *Data Streams: Models and Algorithms*, volume 31. Springer Science & Business Media, 2007.
- [342] 1998 Darpa Intrusion Detection Evaluation Data Set, 1998. URL <https://www.ll.mit.edu/r-d/datasets/1998-darpa-intrusion-detection-evaluation-dataset>, Accessed 28 Jul 2018.
- [343] Albert Bifet. *SEAGenerator.java*, 2015. URL <https://github.com/Waikato/moa/blob/master/moa/src/main/java/moa/streams/generators/SEAGenerator.java>, Accessed 28 Jul 2018.
- [344] Richard Kirkby. *AgrawalGenerator.java*, 2013. URL <https://github.com/Waikato/moa/blob/master/moa/src/main/java/moa/streams/generators/AgrawalGenerator.java>, Accessed 28 Jul 2018.
- [345] Raj Basu, Robert K Cunningham, Seth E Webster, and PR Lippmann. Detecting Low-Profile Probes and Novel Denial-of-Service Attacks. In *Proceedings of the 2001 IEEE Workshop on Information Assurance and Security, United States Military Academy*, pages 5–10, 2001.
- [346] Luis Torgo. *DMwR: Functions and Data for "Data Mining With R"*, 2013. Version 0.4.1, URL <https://CRAN.R-project.org/package=DMwR>, Accessed 21 May 2018.
- [347] Luis Torgo. *Data Mining With R: Learning With Case Studies*. Chapman and Hall/CRC, 2010.
- [348] Ali Shiravi, Hadi Shiravi, Mahbod Tavallaee, and Ali A Ghorbani. *Intrusion Detection Evaluation Dataset (ISCXIDS2012)*, 2012. URL <http://www.unb.ca/cic/datasets/ids.html>, Accessed 27 Jul 2018.
- [349] Wireshark. *TShark*, 1998–2018. Version 1.10.6, URL <https://www.wireshark.org/>, Accessed 27 Jul 2018.
- [350] Shawn Ostermann. *TCPTTRACE*, 1998–2018. Version 6.6.7, URL <http://www.tcptrace.org/>, Accessed 17 Jan 2017.
- [351] Microsoft. *Windows Server 2012 R2 Datacenter*, 2012. URL <https://www.microsoft.com/en-gb/Licensing/product-licensing/windows-server-2012-r2.aspx>, Accessed 17 Jan 2017.
- [352] Security Onion Solutions LLC. *SecurityOnion*, 2009–2017. Version 12.04.5.1-20150205, URL <https://securityonion.net/>, Accessed 17 Jan 2017.
- [353] Larry Wall. *Perl*, 1987–2017. Version 5.18.2, URL <https://www.perl.org/>, Accessed 17 Jan 2017.
- [354] Wenke Lee. *A Data Mining Framework for Constructing Features and Models for Intrusion Detection Systems*. PhD thesis, Graduate School of Arts and Sciences, Columbia University, 1999.

- [355] Wenke Lee, Salvatore J Stolfo, and Kui W Mok. Mining in a Data-Flow Environment: Experience in Network Intrusion Detection. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 114–124. ACM, 1999.
- [356] Wenke Lee and Salvatore J Stolfo. A Framework for Constructing Features and Models for Intrusion Detection Systems. *ACM Transactions on Information and System Security (TISSEC)*, 3 (4): 227–261, 2000.
- [357] Tin Kam Ho. Random Decision Forests. In *proceedings of the 3rd International Conference on Document Analysis and Recognition*, volume 1, pages 278–282. IEEE, 1995.
- [358] Tin Kam Ho. The Random Subspace Method for Constructing Decision Forests . *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20 (8): 832–844, 1998. doi: 10.1109/34.709601.
- [359] Leo Breiman. Random Forests. *Machine Learning*, 45 (1): 5–32, Oct 2001. URL <https://doi.org/10.1023/A:1010933404324>.
- [360] Andy Liaw. *Randomforest: Breiman and Cutler’s Random Forests for Classification and Regression*, 2018. Version 4.6-14, URL <https://CRAN.R-project.org/package=randomForest>, Accessed 21 May 2018.
- [361] Jinbo Bi, Kristin Bennett, Mark Embrechts, Curt Breneman, and Minghu Song. Dimensionality Reduction via Sparse Support Vector Machines. *Journal of Machine Learning Research*, 3 (Mar): 1229–1243, 2003.
- [362] Miron B Kurşa and Witold R Rudnicki. Feature Selection with the Boruta Package. *J Stat Softw*, 36 (11): 1–13, 2010.
- [363] Witold R Rudnicki, Mariusz Wrzesień, and Wiesław Paja. All Relevant Feature Selection Methods and Applications. In *Feature Selection for Data and Pattern Recognition*, pages 11–28. Springer, 2015.
- [364] Bernard L Welch. The Generalization of ‘Student’s’ Problem when Several Different Population Variances are Involved. *Biometrika*, 34 (1/2): 28–35, 1947.
- [365] Graeme D Ruxton. The unequal variance t-test is an underused alternative to Student’s t-test and the Mann–Whitney U test. *Behavioral Ecology*, 17 (4): 688–690, 2006.
- [366] Jason Brownlee. *Blog Post: 8 Tactics to Combat Imbalanced Classes in Your Machine Learning Dataset*, 2015. URL <https://machinelearningmastery.com/tactics-to-combat-imbalanced-classes-in-your-machine-learning-dataset/>, Accessed 16 Jul 2018.
- [367] M Mostafizur Rahman and D Davis. Cluster Based Under-Sampling for Unbalanced Cardiovascular Data. In *Proceedings of the World Congress on Engineering*, volume 3, pages 3–5, 2013.
- [368] Nitesh V Chawla. Data Mining for Imbalanced Datasets: An Overview. In *Data mining and knowledge discovery handbook*, pages 875–886. Springer, 2009.
- [369] M Mostafizur Rahman and DN Davis. Addressing the Class Imbalance Problem in Medical Datasets. *International Journal of Machine Learning and Computing*, 3 (2): 224–228, 2013.
- [370] Juan D Rodríguez, Aritz Pérez, and Jose A Lozano. Sensitivity Analysis of K-Fold Cross Validation in Prediction Error Estimation. *IEEE transactions on pattern analysis and machine intelligence*, 32 (3): 569–575, 2010.

- [371] Jake R Conway, Alexander Lex, and Nils Gehlenborg. UpSetR: an R package for the visualization of intersecting sets and their properties. *Bioinformatics*, 33 (18): 2938–2940, 2017.
- [372] Alexander Lex, Nils Gehlenborg, Hendrik Strobelt, Romain Vuillemot, and Hanspeter Pfister. UpSet: Visualization of Intersecting Sets. *IEEE transactions on visualization and computer graphics*, 20 (12): 1983–1992, 2014.
- [373] Peter B. Nemenyi. Distribution-Free Multiple Comparisons. In *Biometrics*, volume 18, page 263. International Biometric SOC 1441 I ST, NW, Suite 700, Washington, DC 20005-2210, 1962.
- [374] Peter B. Nemenyi. *Distribution-Free Multiple Comparisons*. PhD thesis, Princeton University, 1963.
- [375] Myles Hollander and Douglas A Wolfe. *Nonparametric Statistical Methods - Second Edition*. Wiley-Interscience, 1999.
- [376] Chris Ding and Hanchuan Peng. Minimum Redundancy Feature Selection from Microarray Gene Expression Data. *Journal of bioinformatics and computational biology*, 3 (02): 185–205, 2005.
- [377] Chris Ding and Hanchuan Peng. Minimum Redundancy Feature Selection from Microarray Gene Expression Data . In *Proceedings of the 2003 IEEE Computational Systems Bioinformatics Conference (CSB2003)*, pages 523–528, 2003. doi: 10.1109/CSB.2003.1227396.
- [378] Ken Black. *Business Statistics for Contemporary Decision Making (Sixth Edition)*. John Wiley & Sons, Inc., 2010. ISBN 9780470556672.
- [379] Paul J. Lavrakas. *Encyclopedia of Survey Research Methods*. Sage Publications, Inc., 2008. ISBN 9781412918084.
- [380] Chris J Skinner. Probability Proportional to Size (PPS) Sampling. *Wiley StatsRef: Statistics Reference Online*, pages 1–5, 2014.
- [381] Department of Statistics Online Programs. *Module 6.1: How to Use Stratified Sampling*. In *STAT506: Sampling Theory and Methods*, 2018. URL <https://newonlinecourses.science.psu.edu/stat506/node/2/>, Accessed 30 Jul 2018.
- [382] Mohammad Esfahani Shahrokh and Edward R Dougherty. Effect of Separate Sampling on Classification Accuracy. *Bioinformatics*, 30 (2): 242–250, 2013.
- [383] Sherri L Jackson. *Research Methods and Statistics: A Critical Thinking Approach (Fifth Edition)*. Cengage Learning, 2015. ISBN 9781305257795.
- [384] Research-Methodology. *Cluster Sampling*, 2018. URL <https://research-methodology.net/sampling-in-primary-data-collection/cluster-sampling/>, Accessed 30 Jul 2018.
- [385] Iosif-Viorel Onut. *A fuzzy feature evaluation framework for network intrusion detection*. PhD thesis, Faculty of Computer Science, University of New Brunswick, 2008.
- [386] Stephanie Sundberg. *Cohen's Kappa Statistic*, 2017. URL <http://www.statisticshowto.com/cohens-kappa-statistic/>, Accessed 16 Jul 2018.