University of Reading

Real-Time Pre-Processing Technique for Drift Detection, Feature Tracking, and Feature Selection using Adaptive Micro-Clusters for Data Stream Classification

Thesis Submitted for the Degree of Doctor of Philosophy,

School of Mathematical, Physical, and Computational Sciences

by

Mahmood Shakir Hammoodi

Supervisor: Dr. Frederic Stahl

August 2018

Abstract

Data streams are unbounded, sequential data instances that are generated with high *Velocity*. Data streams arrive online (i.e., instance by instance) and there is no control over the order in which data instances arrive either within a data stream or across data streams. Classifying sequential data instances is a challenging problem in machine learning with applications in network intrusion detection, financial markets and sensor networks. The automatic labelling of unseen instances from the stream in real-time is the main challenge that data stream classification faces. For this, the classifier needs to adapt to concept drifts and can only have a single-pass through the data with a limited amount of memory if the stream is generating data instances at a high *Velocity*. Nowadays the focus of Data Stream Mining (DSM) lies in the development of data mining algorithms rather than on pre-processing techniques. To the best of the author knowledge, at present, there are no developments for truly real-time feature selection in a streaming setting. This research work presents a real-time pre-processing technique, in particular, feature tracking in combination with concept drift detection. The feature tracking is designed to improve DSM classification algorithms by enabling real-time feature selection. The pre-processing technique is based on tracking adaptive statistical summaries of the data and class label distributions, known as Micro-Clusters. Thus the three objectives of this research were to develop a real-time pre-processing technique that can (1) detect a concept drift, (2) identify features that were involved in concept drift and thus potentially change their relevance and (3) build a real-time feature selection method based on the developments mentioned above. The evaluation of the developed technique is based on artificial data streams with known ground truth and real datasets with and without artificially induced concept drift (i.e., controlled and uncontrolled real datasets). It was observed that the developed method for concept drift detection did detect induced concept drifts very well compared with alternative concept drift detection methods. Overall the research represents a first attempt to resolve real-time feature selection for DSM tasks. It has been shown that the technique can indeed identify concept drift, track features, and identify features that may have changed their relevance for the DSM task in real-time. It has also been shown that the developed method for real-time feature selection can improve the accuracy of data stream classification tasks.

Declaration

I confirm that this thesis is my own work. I also declare and certify that, to the best of my knowledge, this thesis does not infringe upon anyone's copyright nor violate any proprietary rights. It is submitted for the purpose of a PhD degree requirement to the School of Mathematical, Physical, and Computational Sciences, the University of Reading, UK. This thesis has not been submitted before for any degree or exam of any other university or institute.

Mahmood Shakir Hammoodi

Certificate

This is to certify that the thesis entitled "**Real-Time Pre-Processing Technique for Drift Detection, Feature Tracking, and Feature Selection using Adaptive Micro-Clusters for Data Stream Classification**" has been prepared under my supervision by Mahmood Shakir Hammoodi for the award of the Degree of Philosophy in Computer Science in the School of Mathematical, Physical, and Computational Sciences, the University of Reading.

Dr. Frederic Stahl
Associate Professor
Department of Computer Science
University of Reading
Polly Vacher Building
Whiteknights
Reading
RG6 6AY

Acknowledgements

First of all, I am thankful to Allah (S.W.T), the All-Mighty, who blessed me with the strength and courage to complete this work.

I would like to express my thanks to the Ministry of Higher Education and Scientific Research of the Republic of Iraq, the Iraqi Cultural Attaché in London, and the University of Babylon for giving me this opportunity.

This thesis has been completed with the help of many people. I am particularly grateful to my supervisor, Associate Professor Dr Frederic Stahl whose moral support and encouragement was essential to finish this piece of work. He was always kind and cooperative. I am also indebted to my colleague Mark Tennant, who has always been a source of inspiration and encouragement.

I also would like to express my thanks to my beloved wife, Mithal, who has offered moral support, encouragement and patient companionship during my study. My thanks are also due to my uncle Abo Wissam, Um Wissam, Wissam, and my children, Hassan and Murtadha. They were the spirit which always encouraged me to continue with my study.

In the end, I acknowledge the role of my family in the accomplishment of this work. The prayers of my parents and the support of my brothers and my sisters who have made all this possible.

And finally, I would like to express my sincere gratitude to all those mentioned above who provided not only much needed time but also their continued support and inspirations which strengthened my pledge to overcome all obstacles in completing this task, and I dedicate this thesis to these people whom I love very much.

Mahmood Shakir Hammoodi

Table of Contents

Page

Abstract **ii**

Declaration **iv**

Certificate **v**

Acknowledgements **vi**

List of Tables **xii**

List of Figures **xix**

List of Abbreviations **xxvii**

1 Introduction **1**

1.1 Background and Problem Statement of the Research 1

1.2 Scope of Research 3

1.3 Motivations for Research 4

1.4 Research Objectives 4

1.5 Methodology of Research 5

1.5.1 Literature Review Stage 5

1.5.2 Design a Framework of the Developed Methods Stage 6

1.5.3 Implementation Stage 7

1.5.4 Evaluation Stage 8

1.6 Organisation of Thesis 8

1.7 Summary 9

2 Background and Literature Review **11**

2.1 Data Stream Mining (DSM) 11
2.1.1 Workflow of DSM 12
2.1.2 Concept Drift 13
2.2 Overview of General Data Stream Processing Techniques 14
2.2.1 Statistical Summaries with Single-Pass Processing 15
2.2.2 Windowing Approaches 15
2.2.3 Adaptive Learning Algorithms 15
2.3 Data Stream Pre-Processing Techniques 16
2.3.1 Minimising the Effect of Feature-Bias 16
2.3.2 Minimising the Effect of Noise 17
2.3.3 Minimising the effect of Outliers 18
2.3.4 Concept Drifts Detection (CDD) Methods 19
2.3.5 Feature Selection (FS) Methods 22
2.4 Adaptive DSM Algorithms 27
2.4.1 Adaptive Data Stream Classification Algorithms 27
2.4.2 Adaptive Data Stream Clustering Algorithms 30
2.5 Summary of Reported Literature 34
2.6 Summary 36

3 Micro-Cluster Nearest Neighbour (MC-NN) 37
3.1 The Structure of MC-NN Micro-Clusters 38
3.2 Absorbing Instances 39
3.3 Splitting of a Micro-Cluster using Variance 41
3.4 Death and Removal of a Micro-Cluster using Triangle Numbers 42
3.5 Taking MC-NN Forward to Develop a Real-time Pre-Processing Technique . . 44
3.6 Experimental Setup 45
3.6.1 Artificial Datasets 45
3.6.2 Real Datasets 46
3.7 Summary 49

4 Real-Time Concept Drift Detection Method using Adaptive Micro-Clusters 50
4.1 Detecting Concept Drift using Adaptive Micro-Clusters 50
4.2 Worked Example 52

4.3 Empirical Evaluation of Real-Time Concept Drift Detection Method 53
4.4 Conclusion 61

5 Real-Time Feature Tracking Method using Adaptive Micro-Clusters 62
5.1 Minimising the Effect of Feature-Bias using Real-Time Min-Max Normalisation 63
5.2 Minimising the Effect of Noise using Low Pass Filter (LPF) 63
5.3 Real-Time Feature Tracking 67
5.3.1 Velocity of Features 67
5.3.2 Measuring of the Data Stream Spread using IQR 68
5.3.3 Splitting of a Micro-Cluster using IQR 68
5.3.4 Comparison of Splitting using IQR or Variance 69
5.3.5 First-In-First-Out (FIFO) Queue with SkipList 71
5.4 Empirical Evaluation of Real-Time Feature Tracking Method using Variance and IQR 79
5.5 Conclusion 104

6 Real-Time Feature Selection Method using Adaptive Micro-Clusters 105
6.1 Feature Analysis and Feature Selection 106
6.2 Monitoring and Analysis of Temporarily Irrelevant Features 107
6.3 Worked Example 108
6.4 Empirical Evaluation of Real-Time Feature Selection Method 109
6.5 Conclusion 118

7 Conclusion and Future Works 119
7.1 Summary 119
7.2 Contributions of Research 121
7.3 Limitations of the Study 122
7.4 Future Works 123
7.4.1 Extensions and Improvements 123
7.4.2 Future Directions 123

References 126

Appendix A Percentage Difference of Split and Death Rates with Low Pass Filter Rate 140

A.1 Artificial and Controlled Real Datasets 140
A.2 Uncontrolled Real Datasets 141

Appendix B FIFO Queue Size 144

B.1 FIFO Queue in combination with Percentage Difference of Split and Death Rates 144
B.1.1 Artificial and Controlled Real Datasets 144
B.1.2 Uncontrolled Real Datasets 145
B.2 FIFO Queue in combination with Alpha Rate of LPF 146
B.2.1 Artificial and Controlled Real Datasets 146
B.2.2 Uncontrolled Real Datasets 147

Appendix C Percentage Difference of Information Gain for Real-Time Feature Selection 149

Appendix D Error Threshold for Splitting a Micro-Cluster 152

D.1 Artificial and Controlled Real Datasets 152
D.2 Uncontrolled Real Datasets 153

Appendix E Percentage Number of Features with Maximum Velocity Combined with Variance or IQR 156

E.1 Artificial Datasets 156
E.2 Controlled Real Datasets with 6 Features 158
E.3 Uncontrolled Real Datasets 159

Appendix F Actual Values of Split and Death Rates 161

Appendix G The Effects of Feature-Bias, Noise, and Outliers on Velocity of Features 167

Appendix H Gradual Concept Drift Detection using Artificial Datasets 172

H.1 Experimental Setup of Artificial Datasets 172
H.2 Results 173
H.2.1 Real-Time Concept Detection Method 173
H.2.2 Real-Time Feature Tracking Method using Variance and IQR 175

Appendix I Recurring Concept Drift Detection using Artificial Datasets 179

I.1 Experimental Setup of Artificial Datasets 179

I.2 Results 180
I.2.1 Real-Time Concept Detection Method 180
I.2.2 Real-Time Feature Tracking Method using Variance and IQR 182

List of Tables

Page

Table 2.1 Summary of concept drift detection methods. 22

Table 2.2 Summary of feature selection methods. 27

Table 2.3 Summary of data stream classification algorithms. 29

Table 2.4 Summary of data stream clustering algorithms. 34

Table 2.5 Summary of the reported literature. 36

Table 3.1 The structure of MC-NN Micro-Clusters. 38

Table 3.2 Setup of the artificial datasets. Drifts were generated through the individual data stream generators and by swapping features. 47

Table 3.3 Setup of the real datasets for the controlled set of experiments for concept drift detection and feature tracking. 47

Table 3.4 Setup of real datasets for the uncontrolled set of experiments for concept drift detection, feature tracking and real-time feature selection. 48

Table 4.1 Adaptation to concept drift using the initially developed and other state-of-the-art methods. 58

Table 4.2 Summary of concept drift adaptation experiments referring to Table 4.1. . 59

Table 5.1 Summary of the experimental results with artificial datasets generated with noise levels of 0%, 5%, 15%, 25%, and 35%. The results are reported for the Time 6 which is the time of swapped features. 88

Table 5.2 Summary of the experimental results with artificial datasets generated with noise levels of 0%, 5%, 15%, 25%, and 35%. The results are reported for Time 6 which is the point at which features had been swapped. 97

Table 5.3 Summary of the experimental results with controlled real datasets. The results are reported for the time of drift onset (Table 3.3) which is the time at which features were swapped. 104

Table 6.1 Summary of the results for the experiments using real datasets with Hoeffding Tree Classifier. 114

Table 6.2 Summary of the results for the experiments using real datasets with Naive-Bayes Classifier. 118

Table A.1 Summary of concept drifts adaptation experiments using artificial and controlled real dataset with different values (from 10% to 100%) of *Percentage Difference* of *Split* and *Death* rates in combination with α. The results are reported for the Time 6 which is the time at which features were swapped. Where a first number refers to number of *false positive* detections, whereas a second number (between round brackets) refers to number of *true positive* detections. . 141

Table A.2 Summary of average accuracy achieved using uncontrolled real dataset (*CoverType*) with different values (from 10% to 100%) of *Percentage Difference* of *Split* and *Death* rates in combination with α. 142

Table A.3 Summary of average accuracy achieved using uncontrolled real dataset (*Diabetic Retinopathy Debrecen*) with different values (from 10% to 100%) of *Percentage Difference* of *Split* and *Death* rates in combination with α. 142

Table A.4 Summary of average accuracy achieved using uncontrolled real dataset (*Gesture Phase Segmentation*) with different values (from 10% to 100%) of *Percentage Difference* of *Split* and *Death* rates in combination with α. 142

Table A.5 Summary of average accuracy achieved using uncontrolled real dataset (*Statlog (Landsat Satellite)*) with different values (from 10% to 100%) of *Percentage Difference* of *Split* and *Death* rates in combination with α. 143

Table A.6 Summary of average accuracy achieved using uncontrolled real dataset (*Waveform (with noise)*) with different values (from 10% to 100%) of *Percentage Difference* of *Split* and *Death* rates in combination with α. 143

Table B.1 Summary of concept drifts adaptation experiments using artificial and controlled real dataset with different values (from 10% to 100%) of *Percentage Difference* of *Split* and *Death* rates in combination with different *FIFO*'s sizes (100, 500, and 1000). The results are reported for the Time 6 which is the time at which features were swapped. Where a first number refers to number of *false positive* detections, whereas a second number (between round brackets) refers to number of *true positive* detections. 145

Table B.2 Summary of average accuracy achieved using uncontrolled real dataset (*CoverType*) with different values (from 10% to 100%) of *Percentage Difference* of *Split* and *Death* rates in combination with different *FIFO*'s sizes (100, 500, and 1000). 145

Table B.3 Summary of average accuracy achieved using uncontrolled real dataset (*Diabetic Retinopathy Debrecen*) with different values (from 10% to 100%) of *Percentage Difference* of *Split* and *Death* rates in combination with different *FIFO*'s sizes (100, 500, and 1000). 145

Table B.4 Summary of average accuracy achieved using uncontrolled real dataset (*Gesture Phase Segmentation*) with different values (from 10% to 100%) of *Percentage Difference* of *Split* and *Death* rates in combination with different *FIFO*'s sizes (100, 500, and 1000). 146

Table B.5 Summary of average accuracy achieved using uncontrolled real dataset (*Statlog (Landsat Satellite)*) with different values (from 10% to 100%) of *Percentage Difference* of *Split* and *Death* rates in combination with different *FIFO*'s sizes (100, 500, and 1000). 146

Table B.6 Summary of average accuracy achieved using uncontrolled real dataset (*Waveform (with noise)*) with different values (from 10% to 100%) of *Percentage Difference* of *Split* and *Death* rates in combination with different *FIFO*'s sizes (100, 500, and 1000). 146

Table B.7 Summary of concept drift adaptation experiments using artificial and controlled real dataset with different values (from 10% to 100%) of α rate in combination with different *FIFO*'s sizes (100, 500, and 1000). The results are reported for the Time 6 which is the time at which features were swapped. Where a first number refers to number of *false positive* detections, whereas a second number (between round brackets) refers to number of *true positive* detections. 147

Table B.8 Summary of average accuracy achieved using uncontrolled real dataset (*CoverType*) with different values (from 10% to 100%) of α rate of *LPF* in combination with different *FIFO*'s sizes (100, 500, and 1000). 147

Table B.9 Summary of average accuracy achieved using uncontrolled real dataset (*Diabetic Retinopathy Debrecen*) with different values (from 10% to 100%) of α rate of *LPF* in combination with different *FIFO*'s sizes (100, 500, and 1000). 147

Table B.10 Summary of average accuracy achieved using uncontrolled real dataset (*Gesture Phase Segmentation*) with different values (from 10% to 100%) of α rate of *LPF* in combination with different *FIFO*'s sizes (100, 500, and 1000). . 148

Table B.11 Summary of average accuracy achieved using uncontrolled real dataset (*Statlog (Landsat Satellite)*) with different values (from 10% to 100%) of α rate of *LPF* in combination with different *FIFO*'s sizes (100, 500, and 1000). 148

Table B.12 Summary of average accuracy achieved using uncontrolled real dataset (*Waveform (with noise)*) with different values (from 10% to 100%) of α rate of *LPF* in combination with different *FIFO*'s sizes (100, 500, and 1000). 148

Table C.1 Summary of average accuracy achieved using uncontrolled real dataset (*CoverType*) with different values (from 10% to 100%) of *Percentage Difference* of Information Gain in combination with different *FIFO*'s sizes (100, 500, and 1000). 150

Table C.2 Summary of average accuracy achieved using uncontrolled real dataset (*Diabetic Retinopathy Debrecen*) with different values (from 10% to 100%) of *Percentage Difference* of Information Gain in combination with different *FIFO*'s sizes (100, 500, and 1000). 150

Table C.3 Summary of average accuracy achieved using uncontrolled real dataset (*Gesture Phase Segmentation*) with different values (from 10% to 100%) of *Percentage Difference* of Information Gain in combination with different *FIFO*'s sizes (100, 500, and 1000). 150

Table C.4 Summary of average accuracy achieved using uncontrolled real dataset (*Statlog (Landsat Satellite)*) with different values (from 10% to 100%) of *Percentage Difference* of Information Gain in combination with different *FIFO*'s sizes (100, 500, and 1000). 150

Table C.5 Summary of average accuracy achieved using uncontrolled real dataset (*Waveform (with noise)*) with different values (from 10% to 100%) of *Percentage Difference* of Information Gain in combination with different *FIFO*'s sizes (100, 500, and 1000). 151

Table D.1 Summary of concept drifts adaptation experiments using artificial and controlled real dataset with different values (from 10% to 100%) of *Percentage Difference* of *Split* and *Death* rates in combination with Θ. The results are reported for the Time 6 which is the time at which features were swapped. Where a first number refers to number of *false positive* detections, whereas a second number (between round brackets) refers to number of *true positive* detections. . 153

Table D.2 Summary of average accuracy achieved using uncontrolled real dataset (*CoverType*) with different values (from 10% to 100%) of *Percentage Difference* of *Split* and *Death* rates in combination with Θ. 153

Table D.3 Summary of average accuracy achieved using uncontrolled real dataset (*Diabetic Retinopathy Debrecen*) with different values (from 10% to 100%) of *Percentage Difference* of *Split* and *Death* rates in combination with Θ. 154

Table D.4 Summary of average accuracy achieved using uncontrolled real dataset (*Gesture Phase Segmentation*) with different values (from 10% to 100%) of *Percentage Difference* of *Split* and *Death* rates in combination with Θ. 154

Table D.5 Summary of average accuracy achieved using uncontrolled real dataset (*Statlog (Landsat Satellite)*) with different values (from 10% to 100%) of *Percentage Difference* of *Split* and *Death* rates in combination with Θ. 154

Table D.6 Summary of average accuracy achieved using uncontrolled real dataset (*Waveform (with noise)*) with different values (from 10% to 100%) of *Percentage Difference* of *Split* and *Death* rates in combination with Θ. 155

Table E.1 Summary of the experimental results with artificial datasets. 3 percentage numbers are stated in the table which are 25%, 50%, and 75%. These percentages represent the highest percentage number of features with maximum *Velocity* combined with *Variance*. The original MC-NN with *Variance* was used. The results are reported for the Time 6 which is the time of swapped features. 157

Table E.2 Summary of the experimental results with artificial datasets. 3 percentage numbers are stated in the table which are 25%, 50%, and 75%. These percentages represent the highest percentage number of features with maximum *Velocity* combined with *IQR*. The new MC-NN with *IQR* was used. The results are reported for the Time 6 which is the time of swapped features. 158

Table E.3 Summary of the experimental results with controlled real datasets. 3 percentage numbers are stated in the table which are 25%, 50%, and 75%. These percentages represent the highest percentage number of features with maximum *Velocity* combined with *Variance*. The original MC-NN with *Variance* was used. The results are reported for the Time 6 which is the time of swapped features. . 159

Table E.4 Summary of the experimental results with controlled real datasets. 3 percentage numbers are stated in the table which are 25%, 50%, and 75%. These percentages represent the highest percentage number of features with maximum *Velocity* combined with *IQR*. The new MC-NN with *IQR* was used. The results are reported for the Time 6 which is the time of swapped features. 159

Table E.5 Summary of the experimental results with uncontrolled real datasets. 3 percentage numbers are stated in the table which are 25%, 50%, and 75%. These percentages represent the highest percentage number of features with maximum *Velocity* combined with *IQR*. The results are reported for the average accuracy of the Hoeffding Tree classifier achieved using the developed real-time feature selection method. 160

Table H.1 Setup of the artificial datasets. Drifts were generated by swapping features gradually. 173

Table H.2 Adaptation to concept drift using the initially developed and other state-of-the-art methods. 174
Table H.3 Summary of concept drift adaptation experiments highlighted in Table H.2. 175
Table H.4 Summary of the experimental results with artificial datasets generated. The results are reported for the Time 6 which is the time of swapped features. . 178

Table I.1 Setup of the artificial datasets. Drifts were generated by swapping features. 179
Table I.2 Adaptation to concept drift using the initially developed and other state-of-the-art methods. 181
Table I.3 Summary of concept drift adaptation experiments highlighted in Table I.2. 182
Table I.4 Summary of the experimental results with artificial datasets generated. The results are reported for the Time 6 which is the time of swapped features. . 184

List of Figures

Page

Figure 1.1 An example of Micro-Clusters with two features. 6

Figure 1.2 The framework of the developed methods. 7

Figure 2.1 Workflow of feature selection and concept drift detection with a classifier. 12

Figure 2.2 Workflow of Data Stream Mining. 13

Figure 2.3 Types of concept drifts. 14

Figure 2.4 An example of *IQR*. 19

Figure 3.1 An example of adding a new instance to the nearest Micro-Cluster. . . . 39

Figure 3.2 An example of two features within two Micro-Clusters. 40

Figure 3.3 An example of adding a new instance to the nearest Micro-Cluster. . . . 40

Figure 3.4 An example of creating a new Micro-Cluster. 41

Figure 3.5 Splitting of a Micro-Cluster according to the feature with the highest *Variance*. 42

Figure 3.6 The process of calculating the *Triangle Number* for a Micro-Cluster. . . 43

Figure 3.7 An example of *Triangle Number* calculation. The shaded areas signify the time stamps. Where the Micro-Cluster has participated in absorbing new instances for a specific time stamp. 44

Figure 4.1 An example of Micro-Cluster *Split* and *Death* rate. 51

Figure 4.2 An example of concept drift detection using the Micro-Clusters *Split* and *Death* rates. 52

Figure 4.3 The results of *SEA* Data Stream Generator using Micro-Cluster *Percentage Difference* of *Split* and *Death* rates for drift detection. 54

Figure 4.4 The results of *HyperPlane* Data Stream Generator using Micro-Cluster *Percentage Difference* of *Split* and *Death* rates for drift detection. 55
Figure 4.5 The results of *Random Tree* Data Stream Generator using Micro-Cluster *Percentage Difference* of *Split* and *Death* rates for drift detection. 56
Figure 4.6 The results of *CoverType* Dataset with 6 Features using Micro-Cluster *Percentage Difference* of *Split* and *Death* rates for drift detection. 57
Figure 4.7 The results of *Diabetic Retinopathy Debrecen* Dataset with 6 Features using Micro-Cluster *Percentage Difference* of *Split* and *Death* rates for drift detection. 59
Figure 4.8 The results of *Gesture Phase Segmentation* Dataset with 6 Features using Micro-Cluster *Percentage Difference* of *Split* and *Death* rates for drift detection. 59
Figure 4.9 The results of *Statlog (Landsat Satellite)* Dataset with 6 Features using Micro-Cluster *Percentage Difference* of *Split* and *Death* rates for drift detection. 60
Figure 4.10 The results of *Waveform (with Noise)* Dataset with 6 Features using Micro-Cluster *Percentage Difference* of *Split* and *Death* rates for drift detection. 60

Figure 5.1 Flowchart of Min-Max *Normalisation*. 64
Figure 5.2 An example of Min-Max *Normalisation* in real-time. 65
Figure 5.3 Flowchart of *LPF*. 66
Figure 5.4 An example of *LPF*. 67
Figure 5.5 An example of feature *Velocity*. 68
Figure 5.6 Splitting of a Micro-Cluster with *IQR*. 69
Figure 5.7 An example of splitting a Micro-Cluster with *Variance* and *IQR*. 70
Figure 5.8 Real-time sorting of the feature data values using a *SkipList*. 72
Figure 5.9 An example of the update for the *Quartile Identifiers* for Case 1 with a value inserted left of *Q1*. 73
Figure 5.10 An example of the update for the *Quartile Identifiers* for Case 1 with a value inserted between of *Q1* and *Q3*. 74
Figure 5.11 An example of the update for the *Quartile Identifiers* for Case 1 with a value inserted right of *Q3*. 75
Figure 5.12 An example of the update for the *Quartile Identifiers* for Case 2 with a value inserted left of *Q1*. 76

Figure 5.13 An example of the update for the *Quartile Identifiers* for Case 2 with a value inserted between *Q1* and *Q3*. 76
Figure 5.14 An example of the update for the *Quartile Identifiers* for Case 2 with a value inserted right of *Q3*. 77
Figure 5.15 An example of the update for the *Quartile Identifiers* for Case 3 with a value inserted left of *Q1*. 78
Figure 5.16 An example of the update for the *Quartile Identifiers* for Case 3 with a value inserted between *Q1* and *Q3*. 78
Figure 5.17 An example of the update for the *Quartile Identifiers* for Case 3 with a value inserted right of *Q3*. 79
Figure 5.18 The results of *SEA* data stream generator with a noise level of 0% using Micro-Clusters for tracking features. 80
Figure 5.19 The results of *SEA* data stream generator with a noise level of 5% using Micro-Clusters for tracking features. 80
Figure 5.20 The results of *SEA* data stream generator with a noise level of 15% using Micro-Clusters for tracking features. 81
Figure 5.21 The results of *SEA* data stream generator with a noise level of 25% using Micro-Clusters for tracking features. 81
Figure 5.22 The results of *SEA* data stream generator with a noise level of 35% using Micro-Clusters for tracking features. 82
Figure 5.23 The results of *HyperPlane* data stream generator with a noise level of 0% using Micro-Clusters for tracking features. 82
Figure 5.24 The results of *HyperPlane* data stream generator with a noise level of 5% using Micro-Clusters for tracking features. 83
Figure 5.25 The results of *HyperPlane* data stream generator with a noise level of 15% using Micro-Clusters for tracking features. 83
Figure 5.26 The results of *HyperPlane* data stream generator with a noise level of 25% using Micro-Clusters for tracking features. 84
Figure 5.27 The results of *HyperPlane* data stream generator with a noise level of 35% using Micro-Clusters for tracking features. 84
Figure 5.28 The results of *Random Tree* data stream generator with a noise level of 0% using Micro-Clusters for tracking features. 85

Figure 5.29 The results of *Random Tree* data stream generator with a noise level of 5% using Micro-Clusters for tracking features. 85
Figure 5.30 The results of *Random Tree* data stream generator with a noise level of 15% using Micro-Clusters for tracking features. 86
Figure 5.31 The results of *Random Tree* data stream generator with a noise level of 25% using Micro-Clusters for tracking features. 86
Figure 5.32 The results of *Random Tree* data stream generator with a noise level of 35% using Micro-Clusters for tracking features. 87
Figure 5.33 The results of *SEA* data stream generator with a noise level of 0% using Micro-Clusters for tracking features. 89
Figure 5.34 The results of *SEA* data stream generator with a noise level of 5% using Micro-Clusters for tracking features. 90
Figure 5.35 The results of *SEA* data stream generator with a noise level of 15% using Micro-Clusters for tracking features. 90
Figure 5.36 The results of *SEA* data stream generator with a noise level of 25% using Micro-Clusters for tracking features. 91
Figure 5.37 The results of *SEA* data stream generator with a noise level of 35% using Micro-Clusters for tracking features. 91
Figure 5.38 The results of *HyperPlane* data stream generator with a noise level of 0% using Micro-Clusters for tracking features. 92
Figure 5.39 The results of *HyperPlane* data stream generator with a noise level of 5% using Micro-Clusters for tracking features. 92
Figure 5.40 The results of *HyperPlane* data stream generator with a noise level of 15% using Micro-Clusters for tracking features. 93
Figure 5.41 The results of *HyperPlane* data stream generator with a noise level of 25% using Micro-Clusters for tracking features. 93
Figure 5.42 The results of *HyperPlane* data stream generator with a noise level of 35% using Micro-Clusters for tracking features. 94
Figure 5.43 The results of *Random Tree* data stream generator with a noise level of 0% using Micro-Clusters for tracking features. 94
Figure 5.44 The results of *Random Tree* data stream generator with a noise level of 5% using Micro-Clusters for tracking features. 95

Figure 5.45 The results of *Random Tree* data stream generator with a noise level of 15% using Micro-Clusters for tracking features. 95
Figure 5.46 The results of *Random Tree* data stream generator with a noise level of 25% using Micro-Clusters for tracking features. 96
Figure 5.47 The results of *Random Tree* data stream generator with a noise level of 35% using Micro-Clusters for tracking features. 96
Figure 5.48 The results of *CoverType* dataset with 6 features (2 features swapped) using Micro-Clusters for tracking features. 98
Figure 5.49 The results of *CoverType* dataset with 6 features (4 features swapped) using Micro-Clusters for tracking features. 99
Figure 5.50 The results of *Diabetic Retinopathy Debrecen* dataset with 6 features (2 features swapped) using Micro-Clusters for tracking features. 99
Figure 5.51 The results of *Diabetic Retinopathy Debrecen* dataset with 6 features (4 features swapped) using Micro-Clusters for tracking features. 100
Figure 5.52 The results of *Gesture Phase Segmentation* dataset with 6 features (2 features swapped) using Micro-Clusters for tracking features. 100
Figure 5.53 The results of *Gesture Phase Segmentation* dataset with 6 features (4 features swapped) using Micro-Clusters for tracking features. 101
Figure 5.54 The results of *Statlog (Landsat Satellite)* dataset with 6 features (2 features swapped) using Micro-Clusters for tracking features. 101
Figure 5.55 The results of *Statlog (Landsat Satellite)* dataset with 6 features (4 features swapped) using Micro-Clusters for tracking features. 102
Figure 5.56 The results of *Waveform (with Noise)* dataset with 6 features (2 features swapped) using Micro-Clusters for tracking features. 102
Figure 5.57 The results of *Waveform (with Noise)* dataset with 6 features (4 features swapped) using Micro-Clusters for tracking features. 103

Figure 6.1 Process of feature selection in real-time. 108
Figure 6.2 An example of feature analysis, feature selection, and monitoring of temporarily irrelevant features. Assumed Information Gains are indicated in italics, and actual Information Gain calculations are not in italics. 109
Figure 6.3 The results of *CoverType* dataset using Micro-Clusters for concept drift detection with/without real-time feature selection. 112

Figure 6.4 The results of *Diabetic Retinopathy Debrecen* dataset using Micro-Clusters for concept drift detection with/without real-time feature selection. 112
Figure 6.5 The results of *Gesture Phase Segmentation* dataset using Micro-Clusters for concept drift detection with/without real-time feature selection. 113
Figure 6.6 The results of *Statlog (Landsat Satellite)* dataset using Micro-Clusters for concept drift detection with/without real-time feature selection. 113
Figure 6.7 The results of *Waveform (with Noise)* dataset using Micro-Clusters for concept drift detection with/without real-time feature selection. 114
Figure 6.8 The results of *CoverType* dataset using Micro-Clusters for concept drift detection with/without real-time feature selection. 115
Figure 6.9 The results of *Diabetic Retinopathy Debrecen* dataset using Micro-Clusters for concept drift detection with/without real-time feature selection. 115
Figure 6.10 The results of *Gesture Phase Segmentation* dataset using Micro-Clusters for concept drift detection with/without real-time feature selection. 116
Figure 6.11 The results of *Statlog (Landsat Satellite)* dataset using Micro-Clusters for concept drift detection with/without real-time feature selection. 116
Figure 6.12 The results of *Waveform (with Noise)* dataset using Micro-Clusters for concept drift detection with/without real-time feature selection. 117

Figure 7.1 Workflow of Data Stream Mining for real-time feature extraction. . . . 124
Figure 7.2 Example of important and extreme limits of feature data's range using *IQR*. 125

Figure F.1 The actual values of *Split* and *Death* rates of *SEA* data stream generator for drift detection. 162
Figure F.2 The actual values of *Split* and *Death* rates of *HyperPlane* data stream generator for drift detection. 163
Figure F.3 The actual values of *Split* and *Death* rates of *Random Tree* data stream generator for drift detection. 164
Figure F.4 The actual values of *Split* and *Death* rates of *CoverType* dataset with 6 features for drift detection. 164
Figure F.5 The actual values of *Split* and *Death* rates of *Diabetic Retinopathy Debrecen* dataset with 6 features for drift detection. 165

Figure F.6 The actual values of *Split* and *Death* rates of *Gesture Phase Segmentation* dataset with 6 features for drift detection. 165

Figure F.7 The actual values of *Split* and *Death* rates of *Statlog (Landsat Satellite)* dataset with 6 features for drift detection. 165

Figure F.8 The actual values of *Split* and *Death* rates of *Waveform (with Noise)* dataset with 6 features for drift detection. 166

Figure G.1 The results of *CoverType* dataset with 6 features (2 features swapped) using Min-Max, *LPF*, and *IQR* for minimising the effects of feature-bias, noise, and outliers. 168

Figure G.2 The results of *CoverType* dataset with 6 features (4 features swapped) using Min-Max, *LPF*, and *IQR* for minimising the effects of feature-bias, noise, and outliers. 168

Figure G.3 The results of *Diabetic Retinopathy Debrecen* dataset with 6 features (2 features swapped) using Min-Max, *LPF*, and *IQR* for minimising the effects of feature-bias, noise, and outliers. 168

Figure G.4 The results of *Diabetic Retinopathy Debrecen* dataset with 6 features (4 features swapped) using Min-Max, *LPF*, and *IQR* for minimising the effects of feature-bias, noise, and outliers. 169

Figure G.5 The results of *Gesture Phase Segmentation* dataset with 6 features (2 features swapped) using Min-Max, *LPF*, and *IQR* for minimising the effects of feature-bias, noise, and outliers. 169

Figure G.6 The results of *Gesture Phase Segmentation* dataset with 6 features (4 features swapped) using Min-Max, *LPF*, and *IQR* for minimising the effects of feature-bias, noise, and outliers. 169

Figure G.7 The results of *Statlog (Landsat Satellite)* dataset with 6 features (2 features swapped) using Min-Max, *LPF*, and *IQR* for minimising the effects of feature-bias, noise, and outliers. 170

Figure G.8 The results of *Statlog (Landsat Satellite)* dataset with 6 features (4 features swapped) using Min-Max, *LPF*, and *IQR* for minimising the effects of feature-bias, noise, and outliers. 170

Figure G.9 The results of *Waveform (with Noise)* dataset with 6 features (2 features swapped) using Min-Max, *LPF*, and *IQR* for minimising the effects of feature-bias, noise, and outliers. 170

Figure G.10 The results of *Waveform (with Noise)* dataset with 6 features (4 features swapped) using Min-Max, *LPF*, and *IQR* for minimising the effects of feature-bias, noise, and outliers. 171

Figure H.1 The results of *SEA* data stream generator using Micro-Cluster *Percentage Difference* of *Split* and *Death* rates for drift detection. 173

Figure H.2 The results of *HyperPlane* data stream generator using Micro-Cluster *Percentage Difference* of *Split* and *Death* rates for drift detection. 173

Figure H.3 The results of *Random Tree* data stream generator using Micro-Cluster *Percentage Difference* of *Split* and *Death* rates for drift detection. 174

Figure H.4 The results of *SEA* data stream generator using Micro-Clusters for tracking features. 176

Figure H.5 The results of *HyperPlane* data stream generator using Micro-Clusters for tracking features. 176

Figure H.6 The results of *Random Tree* data stream generator using Micro-Clusters for tracking features. 177

Figure I.1 The results of *SEA* data stream generator using Micro-Cluster *Percentage Difference* of *Split* and *Death* rates for drift detection. 180

Figure I.2 The results of *HyperPlane* data stream generator using Micro-Cluster *Percentage Difference* of *Split* and *Death* rates for drift detection. 180

Figure I.3 The results of *Random Tree* data stream generator using Micro-Cluster *Percentage Difference* of *Split* and *Death* rates for drift detection. 180

Figure I.4 The results of *SEA* data stream generator using Micro-Clusters for tracking features. 182

Figure I.5 The results of *HyperPlane* data stream generator using Micro-Clusters for tracking features. 183

Figure I.6 The results of *Random Tree* data stream generator using Micro-Clusters for tracking features. 183

List of Abbreviations

ADWIN	ADaptive sliding WINdow
CUSUM	CUmulative SUM
DSM	Data Stream Mining
DDM	Drift Detection Method
EDDM	Early Drift Detection Method
EWMA	Exponential Weighted Moving Average
FIFO	First In First Out
FS	Feature Selection
IQR	Interquartile Range
LPF	Low Pass Filter
MC-NN	Micro-Cluster Nearest Neighbour
Min-Max	Minimum-Maximum
MOA	Massive Online Analysis
Q	Quartile

Chapter 1

Introduction

This chapter presents the research background and discusses the importance of developing dynamic feature selection techniques in combination with real-time concept drift detection methods. There is an apparent lack of techniques that can do both in the field of data stream mining. The discussion introduces the main research areas and factors that have significantly affected the Data Stream Mining algorithms. Then the research problem is discussed under academic and methodological perspectives. Next, the scope of research, motivations for research, the research objectives, and methodology of research are presented. The chapter is concluded with an outline of the thesis and summary of the chapter.

1.1 Background and Problem Statement of the Research

Velocity in Big Data Analytics (Ebbers et al., 2013) refers to data that is generated at ultra-high speed and is live-streamed (i.e., a data stream) whereupon the processing and storing of it in real-time constitutes significant challenges to current computational capabilities in computing systems (Babcock et al., 2002; Gaber et al., 2005). Thus, Data Stream Mining (DSM) has been developed. Useful information from a data stream can be extracted using DSM. In other words, DSM is the analysis of unbounded and sequential data instances that are unseen, generated, labelled automatically, and arrive with a high *Velocity* in real-time. Thus in DSM, algorithms need to be capable of learning over a single-pass through the training data (Gaber et al., 2005). The general area of DSM covered by this research is to solve problems associated with lack of pre-processing techniques in the context of data stream classification, which is the prediction of class labels of new instances in the data stream in real-time. Potential applications that need real-time data stream classification techniques are for data streams in the chemical process industry (Kadlec et al., 2009), intrusion detection in telecommunications (Jadhav et al., 2013), credit card fraud detection (Salazar et al., 2012), Internet traffic management and weblog analysis (Gama, 2010), sensor networks, etc.

Data stream classification need not only be able to incrementally learn but also be able to adapt to concept drift over a statistical time window using the most recent data from the stream (Gama and Gaber, 2007; Bifet, 2009; Hoens et al., 2012).

A concept drift occurs if the pattern encoded in the data stream changes. DSM has developed various real-time versions of established predictive data mining algorithms that adapt to concept drift and keep the model accurate over time, such as CVFDT (Hulten et al., 2001) and G-eRules (Le et al., 2017). The benefit of classifier independent concept drift detection methods is that they give information about the dynamics of data generation (Gama et al., 2014). Common drift detection methods are for example ADaptive sliding WINdow (ADWIN) (Bifet and Gavalda, 2007), Drift Detection Method (DDM) (Gama et al., 2004) and the Early Drift Detection Method (EDDM) (Baena-Garcıa et al., 2006).

However, these methods are suffering from feature-bias, outliers, and noise (Dongre and Malik, 2014; Brzezinski and Stefanowski, 2014; Frías-Blanco et al., 2015). In addition, no drift detection method devised to-date can provide potentially highly valuable insights as to which features are involved in the concept drift. For example, if a feature is contributing to a concept drift, it can be assumed that the feature may have become either more or less relevant to the current concept. This has inspired the development of a real-time feature tracking method based on feature contribution information for the purpose of feature selection to identify features that have become relevant or irrelevant due to concept drift. Thus, in this research, a technique for detecting causality of drifts, and providing the feature contribution information over a statistical time window of data instances which is kept in short-term memory in real-time has been developed. Based on this, tracking features and identifying the relevant features of a classifier for the purpose of feature selection in real-time have also been developed.

Feature contribution information in this research is formulated or represented as *Velocity* and the spread of a feature's data. *Velocity* is the rate of change of features *centroids* over a statistical time window (i.e., the difference between the current and previous *centroid* of each feature). Regarding feature selection, common feature selection methods are for example Linear Discriminant Analysis (LDA), Canonical Correlation Analysis (CCA), Multi-View CCA, and

Principal Component Analysis (PCA) (Ahsan and Essa, 2014; Lee et al., 2015). These methods can be applied to a sample of the data stream before commencing the training and adaptation of a data stream classifier. However, this would not account for changes in the relevance of features over time for the classification task at hand due to concept drift which can only be dealt with by re-running the above methods to update the feature rankings to accommodate any drifts. However, this can potentially be an expensive procedure especially if there are many dimensions in the data, hence the rationale for a single-pass method requiring the re-evaluation of only the features whose classification relevance has changed since the last pass.

This research, therefore, describes a concept drift detection method for data stream classification algorithms with the feature tracking information feedforward capability linking features to concept drifts over a statistical time window for feature selection purposes. The method only needs to examine features that have potentially changed their relevance and only when there is an indication that the relevance of a feature may have changed. The developed method can be used with any learning algorithm either as a real-time wrapper or a batch classifier or realised inside a real-time adaptive classifier (Domingos and Hulten, 2000).

1.2 Scope of Research

This study aims to improve the DSM classification algorithms in terms of the classification tasks' accuracy through enabling real-time concept drift detection with automatic dimensionality reduction in specific feature selection. In this study, the focus is given more on both detecting drifts and feature selection with adaptive summaries of the data and class distributions with continuous features, known as Micro-Clusters (Zhang et al., 1996; Aggarwal et al., 2003; Tennant et al., 2017). Micro-Clusters group the data points according to their similarities of characteristics using a distance function such as Euclidean distance. Feature extraction is out of the scope of this study. However, it can be developed by further developing this research regarding identifying and detecting redundant features. Feature-bias, outliers and noise may influence DSM techniques (i.e., clustering and classification). Thus, robustness handling or minimising the effect of feature-bias, outliers, and noise has been taken into consideration in this research study. In addition, the in this research developed technique (i.e., a pre-processing technique) is independent of the data mining algorithm, i.e., this is also out of the scope of this study. However, a classifier can be developed by embedding the developed technique to identify and detecting the best feature subset in real-time.

1.3 Motivations for Research

This section discusses the main factors that motivate the research undertaken and investigated which is the problem of real-time feature selection. Nowadays the focus of DSM lies in the development of data mining algorithms rather than on pre-processing techniques. The feature selection previously proposed are typically applied before a classifier is introduced as they are not designed for data streams as they do not take into consideration that the relevance of a feature for a classification task may change over time. To the best of the author knowledge, at present, there are no developments for truly real-time feature selection in a streaming setting. This is important as features may potentially change their relevance for data mining tasks based on specific measures of relevance such as Information Gain. Thus the three objectives of this research were to develop a real-time pre-processing technique that can detect a concept drift, identify features that were involved in concept drift and thus potentially change their relevance, and build a real-time feature selection method.

1.4 Research Objectives

The objective of this work is to highlight the problems associated with lack of pre-processing techniques to a DSM algorithm in real-time. This will improve the accuracy of the classification task of a data stream classifier.

1. To develop and comparatively evaluate a method to identify a drift point (i.e., concept drift) through tracking the significant changes in the statistical summaries in real-time,

2. To develop and evaluate a method to detect causality of drifts (which is identified by the developed method in Objective 1 above) through providing the historical statistics of each feature for identifying which features were involved in drifting over a statistical time window in real-time,

3. To develop a method to continuously analyse the historical statistics provided by the developed method in Objective 2 above with the purpose of selecting the relevant features for adaptive data stream classifier (i.e., dynamic adjustment feature selection in real-time).

1.5 Methodology of Research

This section illustrates the methodology of this research study which can be outlined as follows: literature review stage, design stage of the developed framework and methods, implementation stage, and evaluation stage.

1.5.1 Literature Review Stage

In this stage, the related concepts to the research study are reviewed which are concept drift, feature selection, data stream classification, and data stream clustering. The review includes the concepts related to the various methods and algorithms previously proposed. The review also identifies initially what needs to be improved in order to make effective data stream pre-processing technique in specific real-time feature selection method. This method does not need to examine the entire feature space. This can be achieved by applying concept drift detection method in combination with feature tracking method, and providing the statistical summaries feedforward capability linking features to concept drifts over a statistical time window. In addition, in this stage, the original MC-NN algorithm which was developed by (Tennant et al., 2017) is summarised as it has been identified as a promising classification approach. MC-NN has originally been developed for predictive data stream analytics using Micro-Clusters which are able to adapt to unexpected changes on the stream. However, this research is not concerned with the classification capabilities of MC-NN but in the behaviour of its underlying model during concept drift. Moving in different directions (i.e., a direction represents a feature) is the original purpose of a Micro-Cluster in order to adapt to concept drift and maintain an accurate model. This has inspired the idea of this research study (i.e., tracking the behaviour of Micro-Clusters) to be modified to detect concept drift, causality of concept drift, and enabling continuous feature selection. Figure 1.1 shows an example of Micro-Clusters with two features (i.e., two directions). A Micro-Cluster can move in one or more directions.

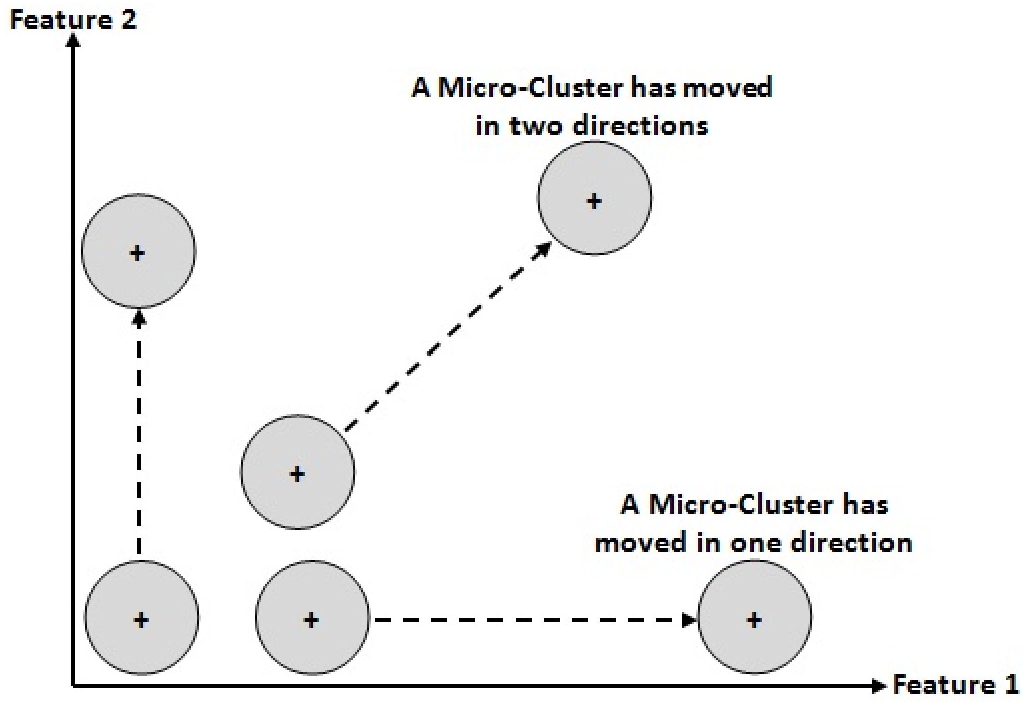

Feature 2
A Micro-Cluster has moved in two directions
A Micro-Cluster has moved in one direction
Feature 1

Figure 1.1: An example of Micro-Clusters with two features.

The review also includes an evaluation which aims to show which method or algorithm previously proposed is capable of handling both concept drift detection and dynamic feature selection, as well as providing the statistical summaries with single-pass processing through streaming data in real-time.

1.5.2 Design a Framework of the Developed Methods Stage

In this stage, three developed methods are identified and designed. A method to identify a drift point (i.e., concept drift) in real-time. A method for detecting causality of drifts providing the historical statistics (i.e., *Velocity* and the spread of a feature's data) of tracked features over a statistical time window in real-time. A method for identifying irrelevant features which are involved in drifting detected in the aforementioned methods through analysing the tracked features and applying dynamic adjustment feature selection in real-time. The developed methods are embedded in a proposed framework which consists of minimising the effect of feature-bias, minimising the effect of noise, drift detection with tracking the involved features, and feature selection. The central components of this framework are drift detection with tracking the involved features and feature selection. The MC-NN approach that has been reviewed in Stage 1.5.1 is used in the central components which detect drifts and provide statistical information for the purpose of tracking which features were involved. The aims and functions of each component are determined, as well as the input and output of each component are identified (Chapters 4, 5, and 6), so that a clear interaction and flow between the components are achieved. Figure 1.2 illustrates the proposed framework.

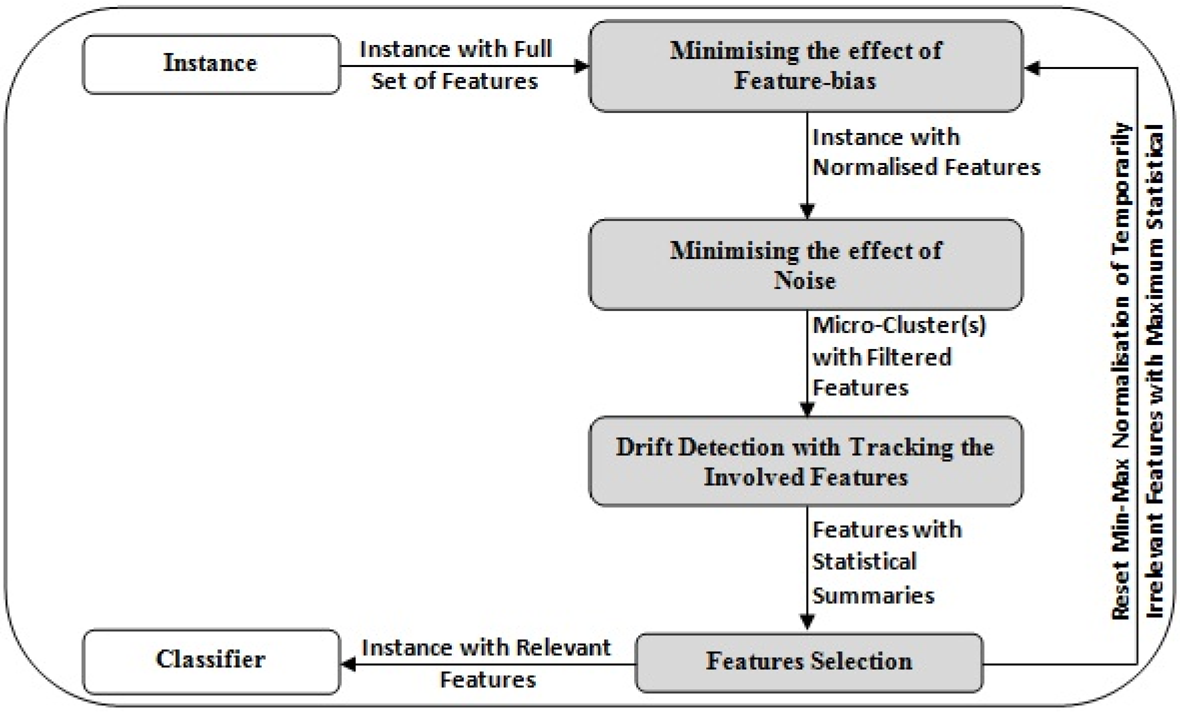

Instance
Instance with Full Set of Features
Minimising the effect of Feature-bias
Instance with Normalised Features
Minimising the effect of Noise
Micro-Cluster(s) with Filtered Features
Drift Detection with Tracking the Involved Features
Features with Statistical Summaries
Features Selection
Instance with Relevant Features
Classifier
Reset Min-Max Normalisation of Temporarily Irrelevant Features with Maximum Statistical

Figure 1.2: The framework of the developed methods.

The major tasks of the proposed framework are to detect a concept drift with the feature tracking information feedforward capability linking features to concept drifts over a statistical time window for feature selection purposes. This method does not need to examine the entire feature space using the adaptive Micro-Clusters.

1.5.3 Implementation Stage

The proposed framework consists of the developed methods that have been designed in stage 1.5.2 is implemented and realised in the Java-based Massive Online Analysis (MOA) framework (Bifet et al., 2010).

Two types of data were used, artificial data stream generators from the MOA framework and real datasets. The reason for using artificial datasets is because MOAs data stream generators enable the introduction of different kinds of concept drift deliberately and thus allow this research to evaluate against a ground truth regarding concept drift.

1.5.4 Evaluation Stage

In this stage, the performance of the developed method for concept drift detection is evaluated with respect to *true positive* detections of known ground truth concept drifts in comparison with the competing alternative concept drift detection methods. Regarding the performance of the developed method for feature tracking is evaluated with respect to *true positive* detections of features involved in drifting. Whereas, the performance of the developed method for real-time feature selection is evaluated with respect to the accuracy of a data stream classifier achieved using the developed method compared with not using the method. Where the data stream classification method chosen was Hoeffding Tree as well as incremental NaiveBayes.

1.6 Organisation of Thesis

The rest of this thesis is organised as follows. **Chapter 2** analyses literature about the fundamental meaning of concept drift, feature selection, data stream classification, data stream clustering, and Micro-Clusters.

Chapter 3 presents the structure of Micro-Cluster Nearest Neighbour (MC-NN) algorithm developed by (Tennant et al., 2017). MC-NN has originally been developed for predictive data stream analytics and is modified in this research to serve as the basis for the methods developed.

Chapter 4 presents the developed method for detecting drifts which is based on the Micro-Cluster structure of the MC-NN classifier presented in Chapter 3. A research paper has been published which consists of the developed method for drift detection with a preliminary result which shows that the developed method is capable of detecting a concept drift but also delivers an indication which features are involved.

- Title: *Towards Online Concept Drift Detection with Feature Selection for data stream classification*,
- Authors: Mahmood Shakir Hammoodi, Frederic Stahl, and Mark Tennant,
- Year: 2016,
- Publisher: IOS Press (*ECAI*).

Chapter 5 presents the developed method for tracking the features involved in drifting by monitoring statistical information provided by adaptive Micro-Clusters. A research paper has been published which consists of the developed method for tracking feature in combination with drift detection as well as preliminary results which show that the developed methods are capable of detecting a concept drift and delivering an indication which features are involved compared with alternative concept drift detection methods.

- Title: *Towards Real-Time Feature Tracking Technique using Adaptive Micro-Clusters*,
- Authors: Mahmood Shakir Hammoodi, Frederic Stahl, Mark Tennant, and Atta Badii,
- Year: 2017,
- Publisher: BCS Specialist Group on Artificial Intelligence.

Chapter 6 presents the developed method for dynamic feature selection in real-time. A research paper has been published which consists of the holistic work of this research study, as well as an in-depth evaluation of the developed methods which are drift detection method, feature tracking method, and real-time feature selection method with respect to different artificial and real datasets.

- Title: *Real-Time Feature Selection Technique with Concept Drift Detection using Adaptive Micro-Clusters for Data Stream Mining*,
- Authors: Mahmood Shakir Hammoodi, Frederic Stahl, and Atta Badii,
- Year: 2018,
- Publisher: Elsevier on Knowledge-Based Systems.

Conclusion and future works as guidelines for further research that can be added to the work drawn from this thesis are summarised in **Chapter 7**.

1.7 Summary

This chapter explains the background of this thesis. It discussed the main factors that have significantly affected the DSM algorithms. This study is motivated by the apparent lack of research in real-time feature selection in a streaming setting. In addition, it is argued that the DSM classification algorithms can be improved in terms of accuracy of classification task by applying a real-time pre-processing technique which detects drifts and selects relevant features for the classifier dynamically. Some backgrounds and concepts of concept drift, feature selection, data stream classification, and data stream clustering will be explained in greater detail in the next chapter.

Chapter 2

Background and Literature Review

In Chapter 1, the aims and objectives were highlighted and described which led to defining the research problem. Chapter 2 presents the background and literature review that formulates the foundation on which this research is based. This chapter discusses the concepts of data stream pre-processing technique, concept drift, feature selection, data stream classification, and data stream clustering. The chapter also reviews related and relevant methods and algorithms. This identifies initially what needs to be improved in order to make an effective data stream pre-processing technique in specific real-time feature selection method. This can be achieved by applying concept drift detection method in combination with feature tracking method. This could be used with any learning algorithm either as a real-time wrapper or a batch classifier or realised inside a real-time adaptive classifier (Domingos and Hulten, 2000). The chapter is organised as follows. In Section 2.1, Data Stream Mining (DSM) is explained. Section 2.2 gives an overview of general data stream processing techniques. The data stream pre-processing techniques are presented in Section 2.3. Adaptive DSM algorithms are discussed in Section 2.4. Section 2.5 analyses the reported methods and algorithms in this chapter with respect to the research objectives. Section 2.6 summarises this chapter.

2.1 Data Stream Mining (DSM)

DSM is the analysis of unbounded and sequential data instances that are unseen and arrive with a high *Velocity* in real-time. In other words, the stream arrives online (i.e., instance by instance) and there is no control over the order in which data instances arrive either within a data stream or across data streams. Once an instance from a data stream has been handled, it can not be retrieved as it is discarded (Gama and Gaber, 2007). The main characteristics of streams include susceptibility to concept drift which is a problem and an area of research as well as handling dimensionality, i.e., features may become irrelevant to a classifier in general over time (Brzeziński, 2010; Ahsan and Essa, 2014; PhridviRaj and GuruRao, 2014). This implies the following requirements on data mining algorithms learning from data streams in real-time:

1. One data instance at a time is examined and processed.
2. A limited amount of memory is required.
3. Predict at any time and demand.
4. Adapt to concept drift in case of changes in the patterns encoded in the data stream.
5. Irrelevant features to a classifier have to be discarded and monitored in real-time, i.e., feature selection needs to be applied dynamically over time.

Next in Section 2.1.1, a general overview of workflow of DSM will be presented.

2.1.1 Workflow of DSM

This section presents the workflow of DSM in terms of feature selection and concept drift detection. Figure 2.1 shows the typical workflow of feature selection and concept drift detection in predictive data stream analysis. The best candidate features from given training data are typically identified and selected by feature selection. The existing feature selection methods need to be applied in advance (i.e., offline) (see Section 2.3.5). A classifier is then introduced to build a model using the selected features which may change over time. Thus, dynamic feature selection is required. Whereas, concept drift detection methods are either applied offline or unable to handle unexpected changes in data (see Section 2.3.4). In addition, in these methods, the features involved in drifting are not identified i.e., the features that cause the drift.

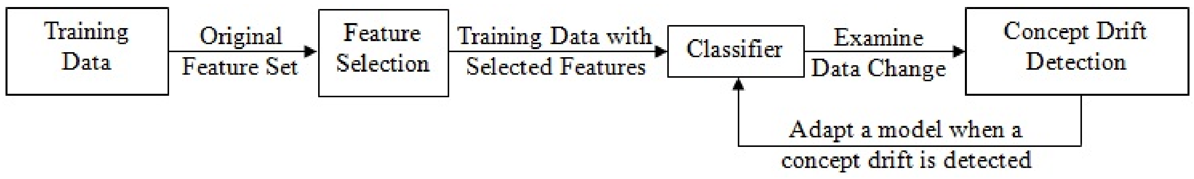


Figure 2.1: Workflow of feature selection and concept drift detection with a classifier.

Thus, adaptive and computationally efficient algorithms are required to analyse streaming data in real-time in terms of concept drift detection with dynamic feature selection which does not to examine the entire feature space, as shown in Figure 2.2. Where Test and Train refer to Test-Then-Train or Prequential. Each individual data instance is used to test the model before it is used for training over time stamps (Bifet and Frank, 2010). Where a time stamp is the current time of a data instance that is used to test and train the model. Examining unexpected change in data (i.e., concept drift) is applied over a statistical windowing approach with a fixed size (i.e., time window with 1000 data instances as an example). Where a time window is a set of time stamps. Hence, features which were involved in drifting are discarded using feature selection (i.e., irrelevant to a classifier). However, the discarded features are ranked and evaluated over time to be selected as relevant to a classifier using feature selection (i.e., dynamic feature selection). Based on this the accuracy of a classification task of a data stream classifier will be improved.

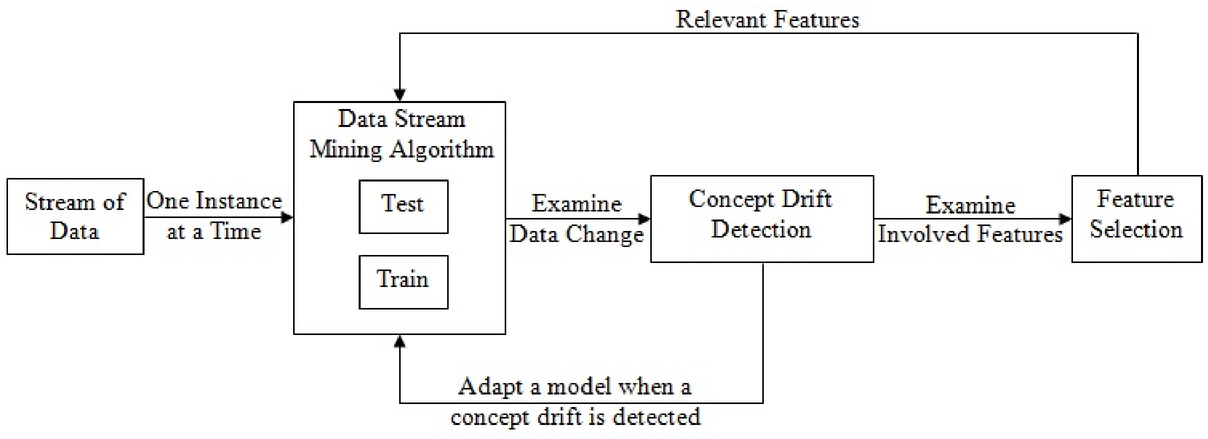


Figure 2.2: Workflow of Data Stream Mining.

Next in Section 2.1.2, a general overview of concept drift will be presented as its the underlying motivation of this research study.

2.1.2 Concept Drift

A concept drift occurs if the pattern encoded in the data stream changes over time. The gathered data changes or shifts after a stability period. Identifying a drift point as distinct from noise or outlier is the first and most challenging task for drift detection algorithms (Bose et al., 2014; Gama et al., 2014). Thus analytics algorithms need to adapt. This issue of concept drift needs to be considered in order to mine relevant data with appropriate accuracy. At least four types of drift can be identified; *gradual*, *sudden*, *recurring*, and *incremental*, as well as noise and outliers which may occur in the data stream (Aggarwal and Yu, 2001; Brzeziński, 2010; Bose et al., 2014; Gama et al., 2014). The different types of concept drift are depicted in Figure 2.3.

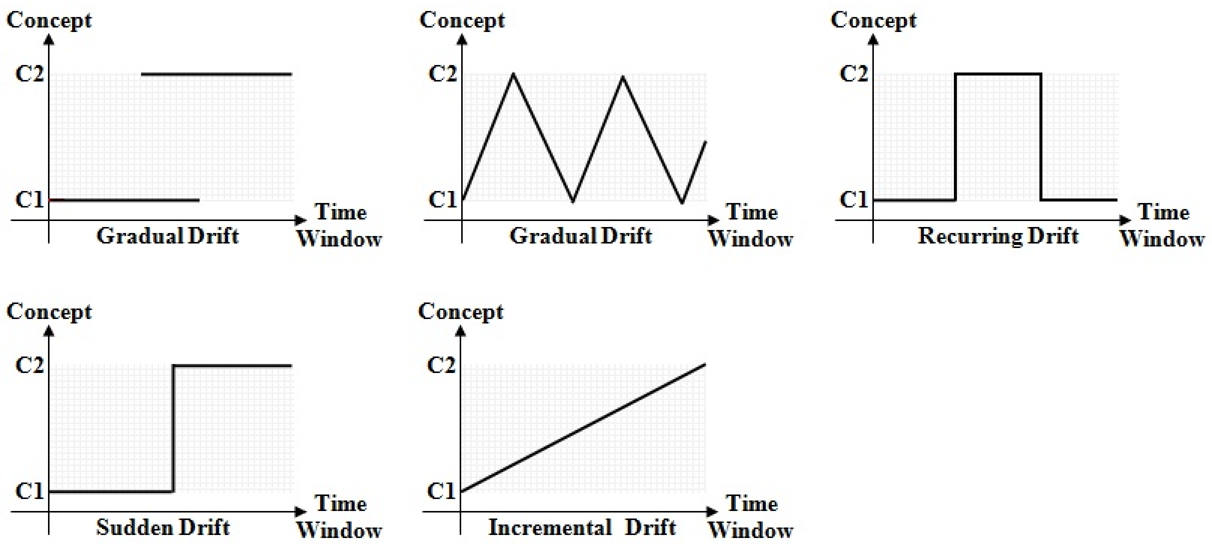


Figure 2.3: Types of concept drifts.

Regarding Figure 2.3 *gradual concept drift* refers to a present concept progressively changing into a new concept within a short period of time whereas *sudden concept drift* refers to the instant replacement of the current concept by a new concept. A *recurring concept drift* refers to a previous concept re-appearing at a later stage, this can be both sudden or gradual. An *incremental concept drift* refers to a constantly evolving concept and these are generally hard to detect. Outliers and noise are not concept drifts. Often it is challenging for concept drift detection methods to distinguish noise and outliers from real concept drift (Brzezinski and Stefanowski, 2014; Gama et al., 2014).

Next in Section 2.2, an overview of general data stream processing techniques is presented, and in Section 2.3 DSM in terms of handling and adapting to concept drift is presented in greater detail.

2.2 Overview of General Data Stream Processing Techniques

This section presents the techniques which are frequently used for processing data streams in terms of handling and adapting to concept drift which are statistical summaries with single-pass processing, windowing approaches, and adaptive algorithms.

2.2.1 Statistical Summaries with Single-Pass Processing

In data stream processing, handling and adapting to unexpected changes (i.e., concept drift) can be achieved by providing and performing statistical summaries over the most recent data instances, and over summarised versions of the old instances. This is called online-offline processing (Ren and Ma, 2009). However, a data stream may be unbounded and arrive continuously at a high *Velocity* over time. This poses challenges as multiple scans over the stream is not possible or infeasible in real-time. Only single-pass processing through the data is required. This will be highlighted in more detail in Section 2.4.

2.2.2 Windowing Approaches

In data streams, the most recent information from the stream is likely to represent the new changes in the data distribution which can be used for the purpose of analysis of concept drift. Windowing approaches have been used to deal with concept drift using only recent data instead of the whole data stream (Gama and Gaber, 2007; Bifet, 2009; Hoens et al., 2012). There are three commonly used models in data streams which are *sliding windows*, *damped windows*, and *landmark windows* (Silva et al., 2013). In the *sliding windows*, only the most recent information from the data stream is kept in a data structure (i.e., a *first in*, *first out* (*FIFO*) queue) whose size can be fixed or identified in advance. The first value added to the window will be the first one to be removed. The *damped windows* provide the most recent information by associating weights with instances from the data stream. The weights of the instances decrease with time, and more recent instances receive a higher weight than older instances. Regarding the *landmark windows*, the whole data stream is divided into chunks (i.e., windows) to be handled as updating units. When a new landmark arrives, all instances saved into the window are deleted, and the new instances from the current landmark are saved in the window until a new landmark arrives. However, in data stream monitoring or analytics, the *sliding window* is frequently used as it is keeping the most recent information of the stream (Zhu and Shasha, 2003; Lakshmi and Reddy, 2015).

2.2.3 Adaptive Learning Algorithms

Adaptive learning algorithms often need to be operated in environments that are changing rapidly or unexpectedly over time. If the data generating process is unstable, the underlying concept may be drifting over time. The ability to incorporate new data is a desirable property of these algorithms and is considered as a natural extension for the incremental learning algorithms (Gama et al., 2014). Adaptive DSM algorithms will be highlighted in more detail in Sections 2.4.1 and 2.4.2.

2.3 Data Stream Pre-Processing Techniques

In general, when applying data mining algorithms, low-quality data will lead to low-quality data mining models. Hence, pre-processing techniques have to be taken into consideration in order to guarantee a quality of the models, and it can often have a significant effect on generalisation performance of a machine learning algorithm (Gaber et al., 2005; Han et al., 2011). Several types of problems can be resolved by applying a data stream pre-processing technique which consists of minimising the effect of feature-bias, noise, and outliers as well as feature subset selection which is the process of identifying and removing irrelevant features which may not be related to the target concept (Hu, 2003; Kotsiantis et al., 2006; Yan et al., 2006; Davis and Clark, 2011). Moreover, in real-world data, the gathered data often changes or shifts after a minimum stability period. A concept drift detection method needs to be considered and taken into consideration in the pre-processing technique in order to mine relevant data with appropriate accuracy (Ramírez-Gallego et al., 2017). This will be highlighted in the next sections.

2.3.1 Minimising the Effect of Feature-Bias

Normalisation is applied to fit the data (i.e., each feature of a new training instance) to be almost distributed in a pre-defined boundary such as [0,100]. *Normalisation* is used to avoid feature-bias which can potentially lead to mis-classifications as the relevant relations between target class labels and features are considered by the classifier to be more or less important than they actually are (Fan and Davidson, 2006; Pelayo Ramirez, 2011).

Three frequently used types of *Normalisation* are i.e. Min-Max (see Equation 2.1), Decimal Scaling (see Equation 2.2), and Z-Score (see Equation 2.3) (Ogasawara et al., 2010).

$$x = \left(\frac{current\ value - \min x}{\max x - \min x} \right) * (\max range - \min range) + \min range \tag{2.1}$$

$$x = \frac{x}{(10^d)} \tag{2.2}$$

$$x = \frac{x - \mu}{\sigma} \tag{2.3}$$

x is a feature value of the new data instance. Min-Max is a simple *Normalisation* technique to fit the data in a pre-defined boundary with a min *range* (default 0) and a max *range* (default 100). Decimal Scaling moves the decimal point of a feature value x depending on its maximum absolute value. Whereby d equals to $MAX(|\ y\ |) < 1$ (i.e., the smallest value of a feature y). Lastly, Z-Score normalises a feature value x according to its corresponding mean μ and standard deviation σ feature values. However, Decimal Scaling and Z-Score rely on the σ and the μ and thus require the buffering of data before *Normalisation* can be applied. Whereas, min x and max x of Min-Max can be re-initialised over time stamps as the buffering of data is not required. It does not rely on the σ and the μ as it can be updated instance by instance. In addition, Liu et al. (2011) and Patro and Sahu (2015) stated that in Min-Max, a complex calculation is not required, and it has better performance in terms of *Normalisation* in comparison with the aforementioned techniques. This has inspired the idea of applying Min-Max in real-time in this research study.

2.3.2 Minimising the Effect of Noise

Noise is defined as a random error or *Variance* in a measured variable, or randomly occurring errors (i.e., data inconsistency, out-of-range values) (Zhang, 2008). In machine learning algorithms, mis-classification could happen as noise is misleading relationships between the features and the class labels (Hickey, 1996; Zhang, 2008). Thus, minimising the effect of noise is required. There are some techniques proposed for the purpose of filtering such as the Kalman filter and Grid-based filter (Bhowmik and Roy, 2007) which require buffering of data before filtering (Arulampalam et al., 2002). In the Kalman filter, the posterior density at every time is Gaussian which is parameterized in terms of μ and Covariance. Whereas, in Grid-based technique, the optimal recursion of the filtered density is provided when the state space is discrete and consists of all relevant information required to describe the data. Where, the state space refers to the Euclidean space in which the variables (i.e., the number of inputs, outputs, and states) on the axes are the state variables which can be expressed as vectors. State variables are variables whose values developed gradually through time. However, there is another technique proposed called Low Pass Filter (*LPF*) (Rosenholtz and Zakhor, 1991; Schall et al., 2005). It does not rely on the σ and the μ as it can be updated instance by instance. This has inspired the idea of applying *LPF* in real-time in this research study. *LPF* is a filter that passes signals (i.e., the *Velocity* of a feature) with a frequency lower than a certain cut-off frequency α and attenuates signals with frequencies higher than the cut-off frequency using Equation 2.4.

$$new\ filter[x] = \alpha * new\ value[x] + (1-\alpha) * old\ filter[x] \tag{2.4}$$

x is a feature value of the new data instance. $new\ filter[x]$ and $old\ filter[x]$ can be re-initialised over time stamps as the buffering of data is not required.

2.3.3 Minimising the effect of Outliers

Outliers are data stream instances outside the expected range of values. In other words, it is an observation that is numerically distant from the rest of the data (Zhang, 2008). Hence, it has a significant effect on statistical analysis (Zhang, 2008). Thus, minimising the effect of outliers is needed. There are some techniques proposed for the purpose of outliers detection such as Distance-based and Density-based (Knox and Ng, 1998; Knorr and Ng, 1999; Ramaswamy et al., 2000; Breunig et al., 2000) which are generally unable to deal with the curse of high dimensionality (Aggarwal and Yu, 2005). However, there is another technique proposed to minimise the effect of outliers which is called Inter Quartile Range (*IQR*). It is considered a more robust method to outliers (Leys et al., 2013; Sunitha et al., 2014). In addition, Wang et al. (2017) stated that in *IQR*, a complex calculation is not required as it is much easier to compute in comparison with the aforementioned techniques. It can be computed separately for each feature of a data instance. This will be highlighted in more detail in Chapter 5.

IQR measures the spread of an ordered set of data (i.e., ascending order) by dividing it into quartiles such as Lower Quartile ($Q1$), *median*, and Upper Quartile ($Q3$). $Q1$ and $Q3$ are the middle numbers of the first and second half of an ordered list of feature values, respectively. *median* is the middle number of an ordered list of data values. *IQR* is the difference between $Q3$ and $Q1$ which is given by Equation 2.5. Where, x denotes the feature. Figure 2.4 shows an example of *IQR*.

$$IQR[x] = Q3 - Q1 \tag{2.5}$$

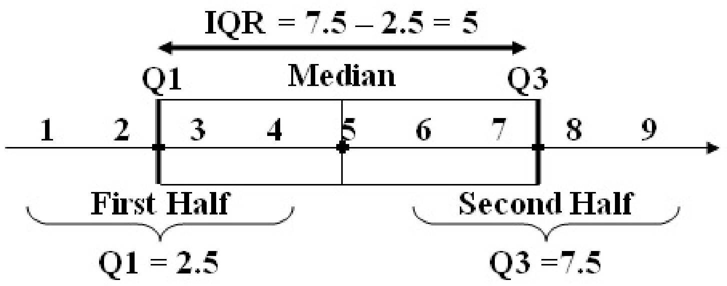


Figure 2.4: An example of *IQR*.

2.3.4 Concept Drifts Detection (CDD) Methods

There exist standalone concept drift detection methods that can be used in combination with batch learning algorithms and a statistical *sliding window* approach (Bifet, 2009). For example some of these methods are CUmulative SUM (CUSUM) (Page, 1954), Drift Detection Method (DDM) (Gama et al., 2004), Early Drift Detection Method (EDDM) (Baena-Garcıa et al., 2006), Exponential Weighted Moving Average (EWMA) (Ross et al., 2012), and ADaptive sliding WINdow (ADWIN) (Bifet and Gavalda, 2007). They are considered as the most well-known drift detection methods that are frequently used (Sidhu and Bhatia, 2015; De Barros et al., 2018). In addition, they are found in an open source DSM framework which is called Massive Online Analysis (MOA) (Bifet et al., 2010).

- **CUSUM** raises alarms when the μ of the input data is significantly different from zero. CUSUM is considered to be 'almost memoryless', however, it can only be applied in one direction, i.e., to detect drifts that only can happen in one direction of the statistics, i.e., only detecting significant increases in feature values.

- **DDM** computes error statistics based on two time windows. An alarm (concept drift) is triggered only for *sudden concept drifts*, while *gradual concept drifts* are not detected (Dongre and Malik, 2014).

- **EDDM** is an extension of DDM estimating a distribution of the distances among classification errors, however, the method is susceptible to noise (Dongre and Malik, 2014; Brzezinski and Stefanowski, 2014).

- **EWMA** is similar to DDM, but the estimate of the error rate is updated faster (i.e., moving averages) for each point in the data stream. Moving averages means generating a set of averages of various subsets of the full dataset. However, the method is susceptible to noise (Frías-Blanco et al., 2015).

- **ADWIN** makes use of a variable size *sliding window*, whereas the size is dependent on observed changes. If there is a change then the window size decreases, otherwise it increases. A problem with ADWIN is that windows can become potentially very large and thus the time to adapt a classifier may increase considerably (Brzeziński, 2010).

Additional improvements to the methods mentioned above have been made with the development of further concept drift detection. Wang and Abraham (2015) have proposed a framework for earliest detecting concept drifts called Linear Four Rates (LFR) with a minimum number of false alarms and subsequently identifying the data points that belong to the new concepts. In LFR, arriving data in batches is not required. It is independent of the underlying classifier employed. Data between warning and detecting time are stored and extracted to learn a new classifier when the stored data are insufficient or too small to learn. Thus, a waiting time is required.

Du et al. (2014) have proposed an approach for detecting concept drift using Entropy which is monitored dynamically over an adaptive *sliding window* in order to detect concept drift. Entropy measures the change in a given dataset (i.e., from order to disorder). Initially, a classifier is trained online incrementally with the arrival of data over a statistical *sliding window*.

Bose et al. (2014) proposed a framework for detecting concept drift by generating populations from the data streams to be analysed for any changes in data values with fixed size windows. The values of a feature are considered as a time series (i.e., populations). For each time series, two populations (i.e., $time$-1 and current $time$) are matched using Cumulative Frequency Distribution which is the total of a class label frequency and all frequencies in a frequency distribution, μ, and σ.

Frías-Blanco et al. (2015) proposed a family of methods to detect the drifts by monitoring significant changes of μ of the performance of a classifier over time. It is an online drift detector and is an extension of the DDM mentioned above. Two versions are proposed (HDDM_A and HDDM_W). In HDDM_A, the Hoeffding's inequality is used which presumes only independent and bounded random variables through providing an upper bound on the probability which is the sum of independent random variables drifts from its expected values by more than a certain amount. In addition, at each incoming value, the error bound is computed for the purpose of detection of significant changes in the moving averages of streaming values (i.e., bounding moving averages). However, it is considered suitable to detect *sudden concept drift* only (Frías-Blanco et al., 2015). Whereas, HDDM$_W$ is an extension of both DDM and EWMA. It monitors a weighted sum which is updated at each incoming value (i.e., weighted moving averages). However, it is considered suitable to detect *gradual concept drift* only (Frías-Blanco et al., 2015).

Pesaranghader and Viktor (2016) proposed a fast hoeffding drift detection method which is an extension of the HDDM family. A *sliding window* of size s (default 200) is used. Drift is detected when a significant change between a maximum and the most recent probabilities of correct predictions is achieved.

Pesaranghader et al. (2017) proposed a drift detection method which is also an extension of the HDDM family. The prediction results of the most recent instances associated with a *sliding window* are weighted. Two variables are updated simultaneously which are a current weighted average and a maximum weighted average. Hence, a significant change between these two variables results in a concept drift.

Barros et al. (2017) proposed a reactive drift detection method which is also another extension of DDM. In this method, the new development which has been applied for DDM is to forget old instances of very long stable concepts in order to provide prediction errors to affect μ error rate and trigger the drifts efficiently.

Liu et al. (2017) proposed a fuzzy windowing concept drift adaptation method. An overlapping period (i.e., old and new concepts) is kept in order to determine the data instances which belong to various concepts. In addition, the membership of data instances in a concept is progressively assessed using fuzzy set theory (i.e., an instance either belongs or does not belong to the set). Hence, a drift is detected when a significant change between old and new concepts is achieved.

Another group of researchers proposed a method for drift detection such as Wilcoxon Rank Sum Test Drift Detector (WSTD) (De Barros et al., 2018) which is also another extension of DDM and Adaptive Cumulative Windows Model (ACWM) (Sebastião et al., 2017) which is also another extension of the HDDM family.

However, none of the aforementioned methods provides information into the causality of the drift, i.e., which features were involved. In addition, these methods are either applied offline or unable to handle different types of concept drifts such as *recurring concept drift* and *gradual concept drift* as well as feature-bias, outliers, and noise (Brzezinski and Stefanowski, 2014).

Table 2.1 summarises the concept drift detection methods presented in this section. The table shows the author(s), a title of the method, and whether the method applies/provides/handles the issue of single-pass processing, statistical summaries, concept drift detection, and dynamic feature selection.

Table 2.1: Summary of concept drift detection methods.

Author(s)	Method	Single-Pass Processing	Statistical Summaries	Concept Drift Detection	Dynamic Feature Selection
Page (1954)	***CUSUM***	Apply	Not provide	Handle	Not handle
Gama et al. (2004)	***DDM***	Apply	Not provide	Handle	Not handle
Baena-Garcıa et al. (2006)	***EDDM***	Apply	Not provide	Handle	Not handle
Ross et al. (2012)	***EWMA***	Apply	Not provide	Handle	Not handle
Bifet and Gavalda (2007)	***ADWIN***	Apply	Not provide	Handle	Not handle

2.3.5 Feature Selection (FS) Methods

Feature selection is used to convert a high dimensional data into a lower dimension by selecting the relevant features which are considered to be highly correlated with classes for the purpose of data classification as learning good classifiers can be achieved by removing irrelevant features (Janecek et al., 2008; Lavanya and Rani, 2011; Han, 2012; Tang et al., 2014).

Feature selection can be classified into filter, wrapper, and embedded (Kohavi and John, 1997; Liu et al., 2006; Zhao et al., 2008; Beniwal and Arora, 2012).

- **Filter:** In this technique, feature selection is independent of the data mining algorithm to be used to assess the relevance of features. A statistical measure needs to be applied for selecting the relevant features using a pre-processing technique such as Information Gain, Fisher score, and Correlation Filtering. However, in filter approaches, the effects of the selected feature subset on the performance of a classifier are ignored (Hoi et al., 2012; Beniwal and Arora, 2012).

- **Wrapper:** The relevant features can be selected using a learning algorithm by searching the space of the best feature subset and evaluating the performance of a data mining algorithm applied to each feature of this subset using one of the three major sequential heuristics such as forward, backward, and stepwise regression. However, it is considered more computationally expensive than filter approaches (Kohavi and John, 1997; Beniwal and Arora, 2012; Hoi et al., 2012).

- **Embedded:** Handling larger datasets can be realised by combining both filter and wrapper approaches into the model training process. However, Hoi et al. (2012) stated that the selected features (i.e., the relevant features) may not be suitable for other data mining algorithms.

Common methods have been proposed for dimensionality reduction such as Linear Discriminate Analysis (LDA), Canonical Correlation Analysis (CCA), Multi-View CCA, and Principal Component Analysis (PCA) (Ahsan and Essa, 2014; Lee et al., 2015). LDA transforms two groups of class labels into two matrices to be matched together. CCA is a statistical method which aims to search a linear subspace in which the correlation between two sets of class labels is maximised. Multi-View CCA (Lee et al., 2015) is an extension of CCA which searches for a pairwise correlation between all sets of class labels. PCA aims to merge the different feature vectors in a low dimensional space of eigenvectors which keeps their direction unchanged when a linear transformation is applied to them.

Relief (Kira and Rendell, 1992) is another method which has been proposed for feature selection by estimating features weights iteratively according to their ability to discriminate among nearest neighbour features using 1-Nearest Neighbour (1-NN) algorithm which was introduced by (Kuncheva and Jain, 1999). 1-NN assigns each new feature to the class label of its nearest neighbour from a saved labelled set. A feature x is selected randomly with two nearest neighbours of x in order to calculate its weight. One nearest neighbour from the same class label of x and the other from a different class. However, Kononenko (1994) stated that the nearest neighbours are identified in the original feature space which may not be correct or true in the weighted feature space. Therefore, Relief-F, multiclass-Relief, and Iterative-Relief (I-Relief) (Kononenko, 1994; Sun, 2007; Oreski and Klicek, 2015) were proposed, which take relationships among features into account. The nearest neighbours of a feature can be handled as hidden features (i.e., not identified or unknown). Thus, weights of a feature are estimated iteratively until the nearest neighbours are identified.

Additional improvements in the methods mentioned above and techniques (i.e., filter, wrapper, and embedded) have been made with the development of further feature selection. Fong et al. (2016) have proposed a wrapper-based feature selection method applied incrementally. Swarm Search (SS) is used which is a computational method to improve a feature selection process in terms of a given feature ranker called the Coefficient of Variation (CV) which describes the σ of feature values relative to their μ that belong to a certain class label. The value of CV needs to be identified in order to partition the data set into two clusters (i.e., one to be retrained and the other to be removed) using *k*-means (Lloyd, 1982). *k*-means is one of the popular algorithms used for clustering data by segmenting *n* observations into *k* non-overlapped clusters (i.e., a pre-defined number of non-hierarchical clusters) with the nearest μ. A complete pass over all the data points is required in order to measure the distance from the data points to each cluster. In addition, in SS, multiple search agents called Particles who work in parallel are used to search the most optimal feature subset at any time. Although each Particle is attracted towards its own best location in history (i.e., individual best), a Particle has a gradient to move randomly. However, features are selected in advance (i.e., offline). A threshold needs to be identified after CV is being calculated in order to identify which features and how many of them have to be selected.

Lee et al. (2015) have proposed a supervised Multi-View Canonical Correlation Analysis (sMVCCA) for integrating class labels of high dimensional data to produce more easily handled data representations for data stream classification. This approach provides labels as view (i.e., pair-wise correlation of class label together with its feature) during label encoding to choose the most correlated class labels with features (i.e., all pairs) of the data using Spearman correlation. It provides statistical dependence between two class labels (i.e., relative position label of the observations within the features). However, in their approach, a summation of pair-wise correlation over any number of class labels has to be identified in order to optimise weights as sMVCCA can only account for correlation between limited class labels. Therefore, feature selection is not handled properly.

Wang et al. (2014) have proposed a method for Online Feature Selection (OFS). In this method, Online Gradient Descent (OGD), *L*2-normalisation, and Epsilon Greedy (EG) are used. OGD is an algorithm used for training purposes by updating a set of parameters (i.e., weights of features belong to a certain class label) repeatedly to minimise an error function. Whereas, *L*2-normalisation is used for minimising the sum of squares of the differences among the target class labels. In addition, *k*-Nearest Neighbour classifier (KNN) is used as a batch learning classifier. KNN is a non-parametric method which provides a class membership by a majority vote of the *k* most similar data instances. If a training instance is misclassified, then OGD is applied as well as *L*2-normalisation to ensure that the *Normalisation* of a classifier is minimised. Whereas, EG is used for feature selection purposes. EG is an algorithm that randomly selects a feature with an optimal estimated probability which can be calculated by *OverallAccuracy* which equals the number of correct predictions divided by total attempts. However, in their work, only two class labels are considered and selected randomly before applying OFS as it is unable to handle multi-class labels.

Yadav and Swetapadma (2014) have proposed a classification method with feature selection using Principal Component Analysis (PCA) method to reduce a training dataset into two dimensional (2D) to be trained (i.e., classified) by Artificial Neural Networks (ANN) which is used for classification purposes. ANN is a computational approach based on a large collection of neural units. In PCA, the Variance-Covariance matrix has to be calculated which is a square matrix that comprises of the *Variances* and Covariances associated with several features. *Variance* is a measure of the spread of data in a feature which can be calculated by the average squared deviation from the μ score of feature values. Whereas, Covariance is the measure of the range to which corresponding features from two sets of an ordered data move in the same direction (i.e., correlation of two features). From the Variance-Covariance matrix, the eigenvectors and eigenvalues are calculated. The eigenvectors will be listed in descending order according to eigenvalues. Those with the lowest order are ignored. Thus, a given training dataset will be reduced to a lower dimension (i.e., 2D). However, 2D is the maximum dimension that can be produced from this method.

Wu et al. (2014) have proposed a method for online feature selection called Second-order for Online Feature Selection (SOFS) using Heap binary data structure. Heap means that a root node key is compared with its children and arranged accordingly (minimum or maximum). In this approach, Max-Heap is applied which means that the value of a root node is greater than or equal to either of its children. Max-Heap is used for storing the smallest Covariance of features. A position of a new feature's Covariance can be adjusted in the Heap when this Covariance is changed. If this Covariance is smaller than the Heap limit, then the root of a Heap is replaced by a current item. Otherwise, the corresponding weight will be set to zero (i.e., ignored). Whereas, a feature with unchanged Covariance will not be checked.

Li et al. (2013) proposed a method for online feature selection called Group Feature Selection with Streaming Features (GFSSF) which handles feature selection at both the group and individual feature levels from the features generated and arrived. In the individual level of feature selection, features from the same group are manipulated by searching the best feature subset from the arrived features using Entropy. Each new arrived feature is matched with a target feature to be identified as a relevant, irrelevant, or redundant feature. Whereas, in the group level of feature selection, grouped features have to be matched together using Entropy as well. A group which provides more information is then selected.

Wu et al. (2013) proposed Fast-Online Streaming Feature Selection (Fast-OSFS) algorithm to improve feature selection performance by selecting a small number of relevant features to train a classifier. Fast-OSFS consists of two main parts which are a redundancy analysis and computational cost reduction. In the first part, a Markov blanket is used to remove redundant features. The Markov blanket of a node consists of all the features' data that separate a node from the rest of a network (i.e., Bayesian network) which is the only knowledge required to predict the behaviour of that node. While, in the second part, the computational cost of conditional independence tests is reduced by only taking into account the subsets within the best candidate features that consists of the recently added feature.

Vishwanath et al. (2013) proposed a framework for feature selection called Dimensionality Reduction for Similarity matching and Pruning of time series data streams (DRSP). In the first stage of this framework, partitioning the arrived data into multi groups over a *sliding window* with a fixed size of both windows and groups. Euclidean distance is used to measure the distance between two pairs of data instances of two groups. If the distance greater than a threshold (i.e., user-defined) then those pairs are pruned.

The aforementioned feature selection methods are by no means an exhaustive list of methods. However, all of them have in common that they are typically applied before a classifier is introduced. Thus they are not designed for data streams as they do not take into consideration that the relevance of a feature for a classification task may change over time. Although online feature selection methods have proposed such as OFS, SOFS, GFSSF, and Fast-OSFS, they are assumed that the candidate features are generated dynamically and arrive one feature at a time, and updates the best feature subset (i.e., best candidate features) from the features arrived by handling each feature according to its arrival. Therefore, these methods are not a good solution to be applied on DSM applications which are assumed that data instances are generated dynamically and arrive one data instance at a time. Where a data instance consists of one or more features. Thus, the overall aim of this research study is to develop a real-time feature selection method which can be applied to DSM applications.

Table 2.2 summarises the feature selection methods presented in this section. The table shows the author(s), a title of the method, and whether the method applies/provides/handles the issue of single-pass processing, statistical summaries, concept drift Detection, and dynamic feature selection.

Table 2.2: Summary of feature selection methods.

Author(s)	Method	Single-Pass Processing	Statistical Summaries	Concept Drift Detection	Dynamic Feature Selection
Lee et al. (2015)	***LDA***	Not apply	Not provide	Not handle	Not handle
Lee et al. (2015)	***CCA***	Not apply	Not provide	Not handle	Not handle
Lee et al. (2015)	***Multi-View CCA***	Not apply	Not provide	Not handle	Not handle
Lee et al. (2015)	***PCA***	Not apply	Not provide	Not handle	Not handle
Wang et al. (2014)	***OFS***	Not apply	Not provide	Not handle	Not handle
Wu et al. (2014)	***SOFS***	Not apply	Not provide	Not handle	Not handle
Li et al. (2013)	***GFSSF***	Not apply	Not provide	Not handle	Not handle
Wu et al. (2013)	***Fast-OSFS***	Not apply	Not provide	Not handle	Not handle

2.4 Adaptive DSM Algorithms

This section presents the adaptive DSM algorithms which are categorised into two main categories clustering and classification (Aggarwal, 2007; Gama, 2010) which will be briefly reviewed in the next sections.

2.4.1 Adaptive Data Stream Classification Algorithms

Building a model incrementally is the main task of data stream classification using the most recent data instances for predicting the class labels of unseen examples. Classification techniques can be categorised into five major categories which are *Tree-based classifiers*, *Rule-based classifiers*, *Nearest Neighbour-based classifiers*, *Neural Network-based classifiers*, and *Ensemble-based classifiers* (Beniwal and Arora, 2012; Aggarwal, 2014). Several methods have been proposed for predictive analytics on data streams using the classification categories mentioned above.

- ***Tree-based classifiers-*** a decision tree is generated based on data instances by creating two types of nodes which are the root and the internal roots, and the leaf nodes. Features are associated with the root and the internal roots. Class labels are associated with the leaf nodes. A notable data stream classifier is the Hoeffding Tree family of algorithms. The Hoeffding Tree algorithm by Domingos and Hulten (Domingos and Hulten, 2000) introduces a decision tree incrementally in real-time. The Hoeffding Tree was improved in terms of speed and accuracy by proposing a Very Fast Decision Tree (VFDT) (Hulten et al., 2001). Although achieving high accuracy using a small sample is the main advantage of these algorithms, concept drifting cannot be handled efficiently as a created sub-trees can only expand from the child nodes onwards. Therefore, further improvements have been made with the development of adaptive trees that can alter entire sub-trees using a *sliding window* approach (Gama and Gaber, 2007; Bifet, 2009). The new version of VFDT was termed CVFTD. Where C stands for Concept Drift (Hulten et al., 2001). Loosely speaking, in CVFDT alternative sub-trees can be induced over time and if an alternative sub-tree outperforms (i.e. in terms of accuracy) the current active sub-tree, then the current sub-tree is replaced with the alternative one. However, if sub-trees are growing continuously, then a large amount of memory is required or consumed. In addition, the CVFDT is susceptible to noise, and is unable to handle *recurring concept drifts* (Chu and Zaniolo, 2004; Yi et al., 2016).

- ***Rule-based classifiers-*** a set of IF-THEN rules is used for classification. The feature values are related to the class labels using different intervals of the features. VFDR (Gama et al., 2011) and G-eRules (Le et al., 2017) have been proposed as *Rule-based* data stream classifiers.

- ***Nearest Neighbour-based classifiers-*** they assume that all data instances relate to points in the n-dimensional space. The k-Nearest Neighbours (KNN) to the new point are identified to be used with a weight for determining the class label of the new point. Adaptive Nearest Neighbour Classification for Data Streams (ANNCDS) (Law and Zaniolo, 2005) and a Similarity Search Structure called the Rank Cover Tree (RCT) (Houle and Nett, 2015) have been developed as *Nearest Neighbour-based classifiers* for streaming data.

- ***Neural Network-based classifiers-*** they are a computational approach based on a large collection of neural units which are connected in order to transmit a signal from one to another. Where each unit receives a set of inputs. Effective Pruning of Neural Network Classifier (EPNNClassifier) (Lazarevic and Obradovic, 2001) has been proposed as *Neural Network-based classifier*.

- ***Ensemble-based classifiers-*** in order to obtain better predictive performance, a combination of classifiers is used to generate the models. Ensemble-based classification (EnsembleC) (Wang et al., 2003) and a Scale-free social network method to handle concept drift (SFNClassifier) (Barddal et al., 2014) have been proposed as *Ensemble-based classifiers* for streaming data.

Further data stream classification algorithms have been proposed using the aforementioned classification categories, such as Accelerated Particle Swarm Optimisation (APSO) with Swarm Search (Fong et al., 2016), Online Data Stream Classification with Limited Labels (ODSCLL) (Loo and Marsono, 2015), On Demand Classification of data streams (ODClassification) (Aggarwal et al., 2004b), Online Data Stream Classification with limited labels (ODSClassification) (Loo and Marsono, 2015), and Prototype-based Classification Model (PrototypeCM) (Shao et al., 2014).

Although most of these works have built-in concept drift detection capability, none of these algorithms takes real-time feature selection into consideration. If feature selection is applied at all, then it is mostly at the beginning of the data stream, and it is assumed that the contribution of each feature to the concept remains invariant over time. Where, a classifier is introduced to build a model using the selected features (see Figure 2.1 Section 2.1.1). However, this is an unrealistic assumption. The relevance of features for the concept may change over time, and thus an online feature selection strategy may very well improve the predictive accuracy of the classifier such as Hoeffding Tree Classifier.

Table 2.3 summarises the data stream classification algorithms presented in this section. The table shows the author(s), a title of the algorithm, and whether the algorithm applies/provides/handles the issue of single-pass processing, statistical summaries, concept drift detection, and dynamic feature selection.

Table 2.3: Summary of data stream classification algorithms.

Author(s)	Algorithm	Single-Pass Processing	Statistical Summaries	Concept Drift Detection	Dynamic Feature Selection
Hulten et al. (2001)	***VFDT***	Apply	Not provide	Not handle	Not handle
Hulten et al. (2001)	***CVFDT***	Apply	Not provide	Not handle	Not handle
Gama et al. (2011)	***VFDR***	Apply	Not provide	Not handle	Not handle
Le et al. (2017)	***G-eRules***	Apply	Not provide	Not handle	Not handle
Law and Zaniolo (2005)	***ANNCDS***	Apply	Not provide	Not handle	Not handle
Houle and Nett (2015)	***RCT***	Apply	Not provide	Not handle	Not handle
Lazarevic and Obradovic (2001)	***EPNNClassifier***	Not apply	Not provide	Not handle	Not handle
Wang et al. (2003)	***EnsembleC***	Apply	Not provide	Not handle	Not handle
Barddal et al. (2014)	***SFNClassifier***	Apply	Not provide	Not handle	Not handle
Fong et al. (2016)	***APSO***	Apply	Not provide	Not handle	Not handle
Loo and Marsono (2015)	***ODSCLL***	Apply	Not provide	Not handle	Not handle
Aggarwal et al. (2004b)	***ODClassification***	Apply	Not provide	Not handle	Not handle
Loo and Marsono (2015)	***ODSClassification***	Apply	Not provide	Not handle	Not handle
Shao et al. (2014)	***PrototypeCM***	Apply	Not provide	Not handle	Not handle

2.4.2 Adaptive Data Stream Clustering Algorithms

Clustering is a task of data mining which separates data into clusters (i.e., groups). In a cluster, the data points are more similar to each other than those in different clusters. Clustering techniques can be categorised into five major categories which are *Hierarchical-based clustering*, *Partitioning-based clustering*, *Density-based clustering*, *Grid-based clustering*, and *Model-based clustering* (Amini et al., 2014). Several data stream cluster algorithms with minimum time and memory demands have been proposed over the years using the aforementioned clustering categories. These algorithms typically need only one pass through the data in order to adapt to concept drifts.

- ***Hierarchical-based clustering-*** a given data is grouped into a tree of clusters. Merging or splitting clusters are required. Some algorithms for streaming data here are Balanced Iterative Reducing and Clustering using Hierarchies (BIRCH) (Zhang et al., 1996), E-Stream (Udommanetanakit et al., 2007), ClusTree (Kranen et al., 2009), and HUE-Stream (Meesuksabai et al., 2011). A notable development of these cluster analysis algorithms is the aforementioned BIRCH which builds statistical summaries of the clusters. BIRCH is able to learn concepts incrementally. These statistical summaries consist of Clustering Features (CF) (i.e., Clustering Feature vector) to group the data points. Where $CF = (n, LS, SS)$. LS means a linear sum of a feature's data, while SS means a square sum of a feature's data. CFs are considered suitable for clustering of data streams as it can be initiated dynamically in real-time. These statistical summaries are also often referred to as Micro-Clusters. However, BIRCH does not forget concepts, and thus its ability to adapt to concept drifts is limited (Ding et al., 2015).

  E-Stream is an evolution based stream clustering method. A new data instance arriving can be considered as a separate cluster if it is outside the boundary of the existing clusters. However, outliers may be fused in (i.e., absorbed) by the existing clusters before splitting. An extension of the E-Stream algorithm known as HUE-Stream. Where clusters are merged using a distance function. The nearest cluster of a new data instance is identified using a distance function as well. Whereas, the splitting of clusters is applied using histogram management.

  The ClusTree (Kranen et al., 2009) algorithm mentioned above is an extension of BIRCH. It implements a hierarchical data stream clustering algorithm. Some improvements are applied to Micro-Clusters by updating their measures (i.e., μ and σ) incrementally without visiting the previous (i.e., past) data stream. A Micro-Cluster has to be stored and represented as a hierarchy of CF at multi-levels (i.e., sub-trees) which is updated by inserting new data into the nearest sub-tree using Euclidean distance to measure the distance between two points (i.e., CFs). Whereas, splitting can be applied according to pairwise distances between CFs which did not contribute to the clustering. However, the height of the generated tree can be increased significantly with each split of CFs. Although a single-pass over the stream is required with limited memory usage (Kranen et al., 2009), noise is inserted at leaf level as a new data point (Hassani et al., 2011).

- ***Partitioning-based clustering-*** a particular set of observations is grouped or clustered (i.e., a partition) according to the similarities of their characteristics. Where a partition represents a cluster. A distance function is used to form the clusters. CluStream (Aggarwal et al., 2003), High-dimensional Projected data Stream clustering (HPStream) (Aggarwal et al., 2004a), and Micro-Cluster Nearest Neighbour (MC-NN) (Tennant et al., 2017) have been proposed as *Partitioning-based clustering*.

  An extension of BIRCH aforementioned above is the CluStream algorithm which extends the Micro-Clusters with a time component, enabling the algorithm to forget old and obsolete concepts and to browse through historical models. The structure of Micro-Clusters is: $< CF2^x, CF1^x, CF2^t, CF1^t, n >$. Where $CF2^x$ is a vector with the sum of squares of the features. $CF1^x$ is a vector with the sum of feature values. $CF2^t$ is a vector with the sum of squares of time stamps. $CF1^t$ is a vector with the sum of time stamps. n is the number of data instances in the cluster. CluStream consists of two phases which are online and offline clustering. Summary statistics are acquired from the data stream in the first phase (i.e., online). These statistical summaries are then used for creating the clusters in the second phase on demand (i.e., offline) using the k-means algorithm (Lloyd, 1982). Although Micro-Clusters applied online as a complete pass over the given data is not required, multiple scans of the data are needed in the second phase (i.e., offline). Hence, large data streams may not be handled properly (Mousavi et al., 2015). In addition, CluStream is unable to handle noise and outliers as stated in (Amini et al., 2011; Barddal et al., 2016; Ghesmoune et al., 2016).

  HPStream has been applied in an online-offline processing. The initial clusters are created using k-means algorithm (i.e., an offline process). The set of features associated with each cluster is updated by fading the entire cluster structure using a Fading Cluster Structure (FCS). FCS gives more importance to recent data by minimising the weight of old observations in a data stream over time (Hahsler et al., 2010). However, the algorithm is susceptible to noise (Gao and Zhang, 2013).

A more recent extension of CluStream is the MC-NN algorithm. MC-NN is also built on a further extension of the CluStream's Micro-Clusters, it adds splitting of Micro-Clusters to adapt to concept drift and uses these Micro-Clusters primarily for parallel predictive analytics in real-time. Whereas, Micro-Clusters which are not participating anymore are removed as they are considered old.

- ***Density-based clustering-*** the clusters are formed in terms of dense area. A given cluster is continuously grown until the density in the neighbourhood reaches some threshold. DenStream (Cao et al., 2006) has been proposed as *Density-based clustering*. The Micro-Clusters were improved in terms of handling noise and outliers by proposing the DenStream which consists of three main tasks which are creating new Micro-Clusters, merging two nearest Micro-Clusters, and deleting Micro-Clusters. A new Micro-Cluster is created when the newly arrived instances are outside the density boundary of nearest Micro-Clusters. Whereas, a Micro-Cluster which is not participating anymore is removed as it is considered an outlier. However, Amini et al. (2014) and Thoriya and Shukla (2015) stated that identifying and removing the outlier Micro-Clusters is a time-consuming process in the algorithm which is the main challenge that this algorithm faces.

- ***Grid-based clustering-*** a given data is divided into a number of data points. Where a data point represents a data instance. The density of each point is calculated, and then cluster centres are identified for the purpose of clustering the nearest points. A Grid-based Clustering algorithm to cluster High-dimensional Data Streams called (GCHDS) (Lu et al., 2005), and Probability and Distribution-based Clustering (POD-Clus) (Chaovalit and Gangopadhyay, 2009) have been proposed as *Grid-based clustering*.

  In GCHDS, a summary of data is generated using a grid structure which is updated over time. The data distributions are analysed on each data point in order to identify the best data points which can be used to construct a subspace in which the clustering process is performed.

  Regarding POD-Clus, the main aim of this algorithm is to update summaries of the data points using the normal distribution. In each POD-Clus' cluster, n data points are received from each incoming data stream. When new data arrives, statistical summaries are calculated such as μ, σ, and the Covariance matrix. POD-Clus measures the similarity between data streams using these summaries. However, Bones et al. (2015) and Shukla et al. (2017) stated that POD-Clus is consider computationally inefficient algorithm.

- ***Model-based clustering-*** the main task of this clustering technique is to recover the original model from data as it assumes that the data are created through a model. Hence, clusters and a relationship of objects to clusters are identified from the recovered model. Distributed data stream clustering called (CluDistream) (Zhou et al., 2007) and incremental and adaptive clustering stream data over *sliding window* called (SWEM) (Dang et al., 2009) have been proposed as *Model-based clustering*.

  In CluDistream, an iterative technique called Expectation Maximization (EM) is used for clustering streaming data. EM identifies maximum estimates (i.e., probabilities) of parameters in statistical models. However, a *landmark window* is used with the EM which is implemented at every node of the distributed network.

  Regarding SWEM, a *sliding window* is used with the aforementioned EM above. SWEM creates a summary of data by scanning the data. Data clusters are then created using the summary.

However, the research presented in this work is inspired by the ability of MC-NN's Micro-Clusters to adapt to concept drift and has developed a new Micro-Cluster structure. This could be used for the purpose of concept drift detection and tracking features to identify the causality of drifting. Thus MC-NN will be discussed in more detail in Chapter 3.

Table 2.4 summarises the data stream clustering algorithms presented in this section. The table shows the author(s), a title of the algorithm, and whether the algorithm applies/provides/handles the issue of single-pass processing, statistical summaries, concept drift detection, and dynamic feature selection.

Table 2.4: Summary of data stream clustering algorithms.

Author(s)	Algorithm	Single-Pass Processing	Statistical Summaries	Concept Drift Detection	Dynamic Feature Selection
Zhang et al. (1996)	***BIRCH***	Apply	Provide	Not handle	Not handle
Udommanetanakit et al. (2007)	***E-Stream***	Not apply	Provide	Not handle	Not handle
Kranen et al. (2009)	***ClusTree***	Not apply	Provide	Not handle	Not handle
Meesuksabai et al. (2011)	***HUE-Stream***	Not apply	Provide	Not handle	Not handle
Aggarwal et al. (2003)	***CluStream***	Not apply	Provide	Not handle	Not handle
Aggarwal et al. (2004a)	***HPStream***	Not apply	Provide	Not handle	Not handle
Tennant et al. (2017)	***MC-NN***	Apply	Provide	Not handle	Not handle
Cao et al. (2006)	***DenStream***	Not apply	Provide	Not handle	Not handle
Lu et al. (2005)	***GCHDS***	Apply	Provide	Not handle	Not handle
Chaovalit and Gangopadhyay (2009)	***POD-Clus***	Apply	Provide	Not handle	Not handle
Zhou et al. (2007)	***CluDistream***	Apply	Provide	Not handle	Not handle
Dang et al. (2009)	***SWEM***	Apply	Provide	Not handle	Not handle

2.5 Summary of Reported Literature

This section analyses the reported methods and algorithms in this chapter with respect to provide the statistical summaries with single-pass processing, adaptation to concept drift in real-time, and handling dynamic feature selection. In the reported literature, some of the proposed algorithms have been applied in online-offline processing such as CluStream, ClusTree, DenStream, E-Stream, and HPStream (Ghesmoune et al., 2016). In data stream analytics, single-pass processing is required (Gaber et al., 2005). Although most of the *grid-based clustering* algorithms and the *model-based clustering* algorithms are able to provide statistical summaries in single-pass processing, the *grid-based clustering* algorithms are consider computationally expensive as stated in (Kriegel et al., 2009), whereas the *model-based clustering* algorithms are susceptible to noise and outliers as stated in (Amini et al., 2014). Regarding concept drift detection methods such as CUSUM, DDM, EDDM, EWMA, and ADWIN, these methods are frequently used (Sidhu and Bhatia, 2015). However, they are either applied offline or unable to handle different types of concept drifts such as *recurring concept drift* and *gradual concept drift* as well as feature-bias, outliers, and noise (Brzezinski and Stefanowski, 2014). In addition, information (i.e., tracking features) into the causes of the drift, i.e. which features were involved, not included or mentioned with these methods. It is possible that features may become more or less relevant for a data mining model, i.e. a decision tree. Thus, feature selection methods are required. In the reported literature, some methods have been proposed such as LDA, CCA, Multi-View CCA, and PCA. Although they are frequently used for the purpose of feature selection (Ahsan and Essa, 2014; Lee et al., 2015), they need to be applied in advance (i.e., offline).

Online feature selection methods have been proposed such as OFS, SOFS, GFSSF, and Fast-OSFS. However, they are assumed that the candidate features are generated dynamically and arrive one feature at a time. This is called streaming feature selection (Wu et al., 2010; Yu et al., 2012) which is out of the scope of this research study. Where DSM applications are assumed that data instances are generated dynamically and arrive one data instance at a time. A data instance consists of one or more features.

This chapter has presented 39 previous methods/algorithms, and Table 2.5 reports the number of methods/algorithms applying/providing/handling the issues that are addressed in this research study. From the literature reported in this chapter, 61.54% of the research works focused on the single-pass processing, 30.77% of the research works concentrate on the statistical summaries, 12.82% of the research works focused on the concept drift detection, while none of the research works handles the dynamic feature selection. Adaptive and computationally efficient feature selection method in combination with concept drift detection method is required as the feature may have become either more or less relevant to the current concept. Where features which were involved in drifting are discarded using feature selection. However, the discarded features are ranked and evaluated over time to be selected as relevant for the classification task using feature selection, i.e., dynamic feature selection which was described previously in this chapter (see Figure 2.2). This can be applied using the statistical summaries provided by MC-NN Micro-Clusters. MC-NN has been applied in single-pass processing using adaptive Micro-Clusters which can adapt to concept drift in real-time. Two main tasks are rapidly applied over time stamps when new concepts arrive which are splitting and removal of Micro-Clusters. Hence, moving in different directions at different speed rates are expected. Where a direction represents a feature. Splitting, removal, and movement of Micro-Clusters could be tracked through providing the statistical summaries over time windows. This could be used for the purpose of concept drift detection and tracking features to identify the causality of drifting. Therefore in this research study, the focus is given more on MC-NN as it is considered as a promising approach rather than the aforementioned methods. MC-NN will be highlighted in greater detail in Chapter 3.

Table 2.5: Summary of the reported literature.

Issue	No. of Previous Methods/Algorithms that Apply/Provide/Handle the Issue	%
Single-Pass Processing	24	61.54%
Statistical Summaries	12	30.77%
Concept Drift Detection	5	12.82%
Dynamic Feature Selection	0	0.00%

2.6 Summary

This chapter provides some background and concepts of data stream pre-processing technique, concept drift, feature selection, data stream classification, and data stream clustering, as well as presents and explains methods and algorithms previously proposed to detect drift and select relevant features. This research aims to use the knowledge about a feature's involvement in the concept drift in order to develop a real-time feature selection method that can be used by a range of classification approaches as it feeds forward information about the involvement of individual features in the drift. This can be achieved using the statistical summaries provided by MC-NN Micro-Clusters. Therefore, this research study is giving focus more on MC-NN rather than the aforementioned methods and algorithms presented in this chapter because of these useful properties of MC-NN. This will be introduced in the next chapter.

Chapter 3

Micro-Cluster Nearest Neighbour (MC-NN)

This chapter analyses and summarises the previously briefly discussed MC-NN algorithm (see Chapter 2) developed by Tennant et al. (2017). MC-NN has been identified in Chapter 2 as a promising data stream classification approach exhibiting several properties that can be adapted to realise real-time feature selection from streaming data. It has originally been developed for predictive data stream analytics using Micro-Clusters which are able to adapt to unexpected changes on the stream through parallelisation. However, the underlying Micro-Cluster structure of MC-NN has been adapted and extended in this research in order to develop a drift detection method. MC-NN generates and updates Micro-Clusters incrementally over time stamps. Where maximum number of Micro-Clusters can be identified by a user (i.e., 100 Micro-Clusters). Moving in different directions (i.e., a direction represents a feature) is the main feature of a Micro-Cluster (i.e., *Velocity*). This can be used for the purpose of tracking and identifying the causality of drifting through tracking statistical summaries (i.e., historical statistics such as *Velocity*) over time windows. Thus MC-NN is discussed in greater detail. Essentially there are three operations to adapt MC-NN to concept drifts: (1) absorption of data instances into nearest Micro-Clusters, (2) splitting of Micro-Clusters with high *Variance* and (3) removal of obsolete Micro-Clusters. The chapter is organised as follows. In Section 3.1, the structure of MC-NN Micro-Clusters is presented. In Section 3.2, absorbing instances is explained. In Section 3.3, splitting of a Micro-Cluster using *Variance* is presented. In Section 3.4, removal of a Micro-Cluster is explained. Section 3.5 discusses as to how MC-NN can be developed for drift detection, feature tracking, and feature selection. Section 3.6 discusses the experimental setup. Section 3.7 summarises this chapter.

3.1 The Structure of MC-NN Micro-Clusters

MC-NN's Micro-Clusters provide statistical summaries of feature values retrieved from the stream over time windows. In Tennant et al. (2017), the structure of Micro-Clusters is: $< CF2^x, CF1^x, CF1^t, n, CL, \varepsilon, \Theta, \alpha, \Omega >$. Details about the Micro-Cluster structure components are listed in Table 3.1.

Table 3.1: The structure of MC-NN Micro-Clusters.

Structure Component	**Description**
$CF2^x$	a vector with the sum of squares of the features
$CF1^x$	a vector with the sum of feature values
$CF1^t$	a vector with the sum of time stamps
n	the number of data instances in the cluster
CL	the majority class label of the cluster
ε	the error count
Θ	the error threshold for splitting the Micro-Cluster
α	the initial time stamp
Ω	a minimum (user defined) threshold for the Micro-Cluster minimum participation

The components listed in Table 3.1 can be used to compute the *centroid* for the feature within a Micro-Cluster x:

$$centroid(x) = \frac{CF1^x}{n} \tag{3.1}$$

and the *Variance* (i.e., the boundary) of a feature within a Micro-Cluster:

$$Variance[x] = \sqrt{\left(\frac{CF2^x}{n}\right) - \left(\frac{CF1^x}{n}\right)^2} \tag{3.2}$$

Equations 3.1 and 3.2 represent the summary information of each feature within a Micro-Cluster which are required for the purpose of absorbing of a new instance to its nearest Micro-Cluster. Hence, the equations are updated after each absorbing within a statistical time window (i.e., this will be highlighted in Section 3.2). Thus, the *centroid* (i.e., the position) and the *Variance* of each feature within the Micro-Cluster are expected to be changed over time stamps. Therefore, Micro-Clusters are moving in different directions (i.e., a direction represents a feature). This can be tracked (i.e., the movement of the Micro-Clusters) for the purpose of identifying the involved features, which features were involved in the drift.

3.2 Absorbing Instances

Each *centroid* of a feature within a Micro-Cluster is updated by adding a new instance to its nearest Micro-Cluster by updating the statistical summaries (see Table 3.1) if it is within the boundary (*Variance*) of the Micro-Cluster (i.e., this is illustrated in Figure 3.1). Where Euclidean distance is used to measure the distance between a new data instance and its nearest Micro-Cluster. The error ε is decremented by 1 if a new instance matches the *CL*. Otherwise, if the nearest Micro-Cluster does not match the *CL*, then the new instance is still added to the nearest Micro-Cluster. However, the Micro-Cluster error ε is incremented by one; also the error ε of the nearest Micro-Cluster that matches the *CL* is incremented by 1. In the case that the data instance is outside the boundary (*Variance*) of its nearest Micro-Cluster, then loosely speaking, the instance builds a new Micro-Cluster. In this chapter, only two features (i.e., dimensions) are displayed for readability of the figures in this thesis. However, Micro-Clusters can be initialised and used with any number of features.

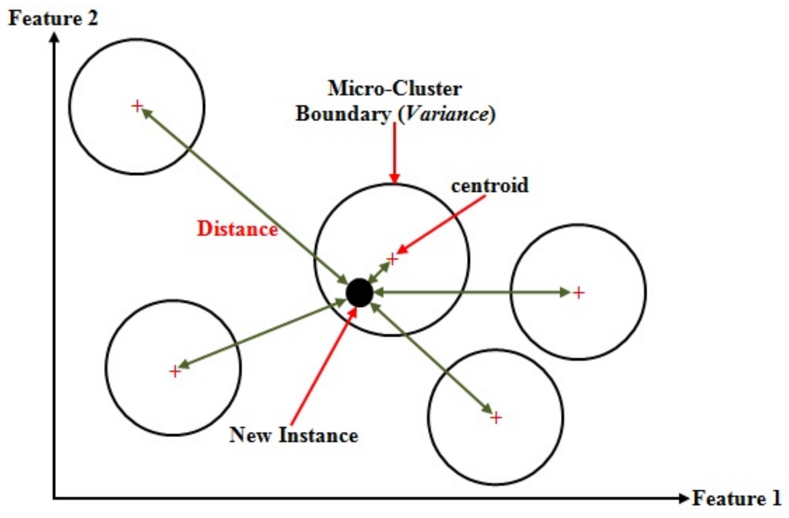


Figure 3.1: An example of adding a new instance to the nearest Micro-Cluster.

Consider the *centroids* of two features within two Micro-Clusters A and B which are shown in Figure 3.2 with *Variance* 941.98 and 1277.65, consequently. Where *centroid* of Feature 1 and 2 within Micro-Cluster A is (3,3), and *centroid* of Feature 1 and 2 within the Micro-Cluster B is (10,8).

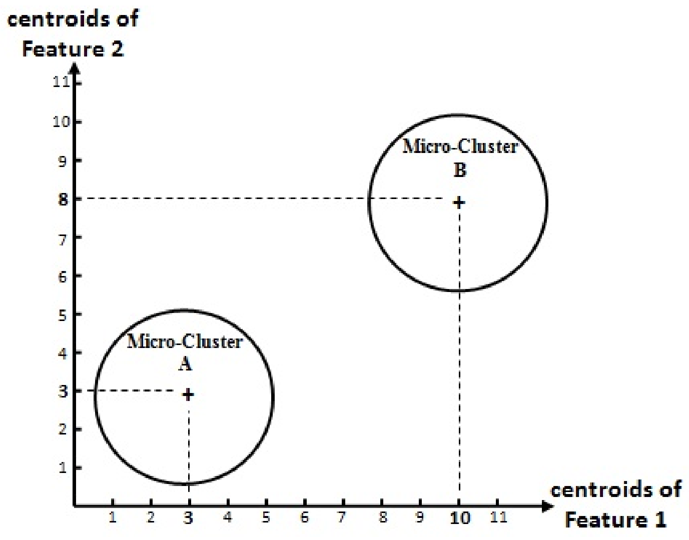


Figure 3.2: An example of two features within two Micro-Clusters.

- ***Case 1*: Consider a new instance (9.5,7) with *Variance* equals to 300.95.**

  Euclidean distance between the new instance and the Micro-Cluster A = $\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2}$ = $\sqrt{(9.5-3)^2+(7-3)^2}$ = $\sqrt{42.25+16}$ = 7.63216.

  Euclidean distance between the new instance and the Micro-Cluster B = $\sqrt{\sum_{i=1}^{n}(x_i-y_i)^2}$ = $\sqrt{(9.5-10)^2+(7-8)^2}$ = $\sqrt{0.25+1}$ = 1.11803.

  As it can be seen that the Micro-Cluster B is the nearest Micro-Cluster to the new instance. The new instance is also inside the boundary (*Variance*) of its nearest Micro-Cluster B. Hence, the new instance is absorbed to its nearest Micro-Cluster B, as shown in Figure 3.3.

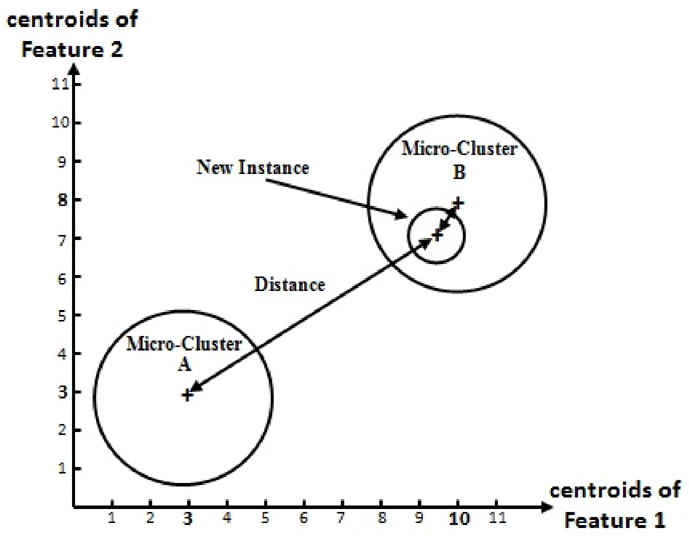


Figure 3.3: An example of adding a new instance to the nearest Micro-Cluster.

- ***Case 2*: Consider a new instance (8,3) with *Variance* equals to 1200.98.**

  Euclidean distance between the new instance and the Micro-Cluster A = $\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$ = $\sqrt{(8-3)^2 + (3-3)^2}$ = $\sqrt{25+0}$ = 5.

  Euclidean distance between the new instance and the Micro-Cluster B = $\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$ = $\sqrt{(8-10)^2 + (3-8)^2}$ = $\sqrt{4+25}$ = 5.38516.

  As it can be seen that the Micro-Cluster A is the nearest Micro-Cluster to the new instance. However, the new instance is not absorbed to its nearest Micro-Cluster A as the new instance is outside the boundary (*Variance*) of its nearest Micro-Cluster. Hence, a new Micro-Cluster is created, the instance becomes a new Micro-Cluster, as shown in Figure 3.4.

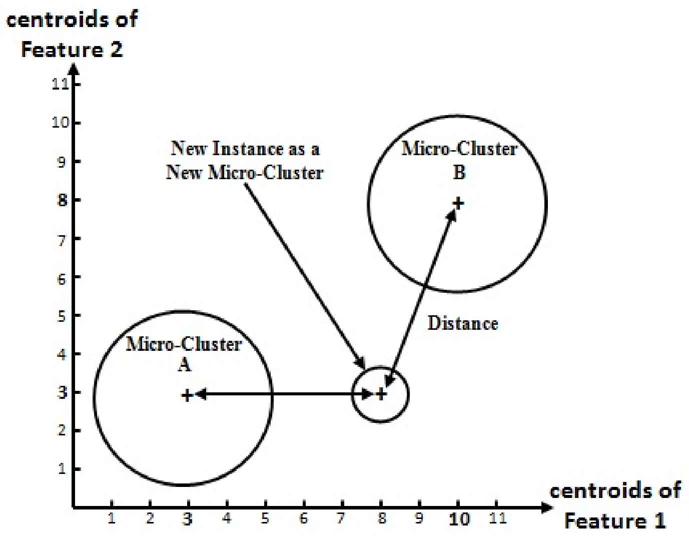


Figure 3.4: An example of creating a new Micro-Cluster.

3.3 Splitting of a Micro-Cluster using Variance

MC-NN splits a Micro-Cluster into two new clusters once the error count ε reaches Θ, and the original Micro-Cluster is removed in order to improve the fit to evolving data streams as it has recorded high rate of false positive participating (i.e., ε). The new Micro-Clusters are placed about the original Micro-Cluster feature that has the greatest *Variance* for a feature x, this is illustrated in Figure 3.5. The assumption MC-NN makes here is that the feature with the highest *Variance* is the most likely to contribute to mis-absorption. However, an example of splitting of a Micro-Cluster can be found in Chapter 5 Section 5.3.4.

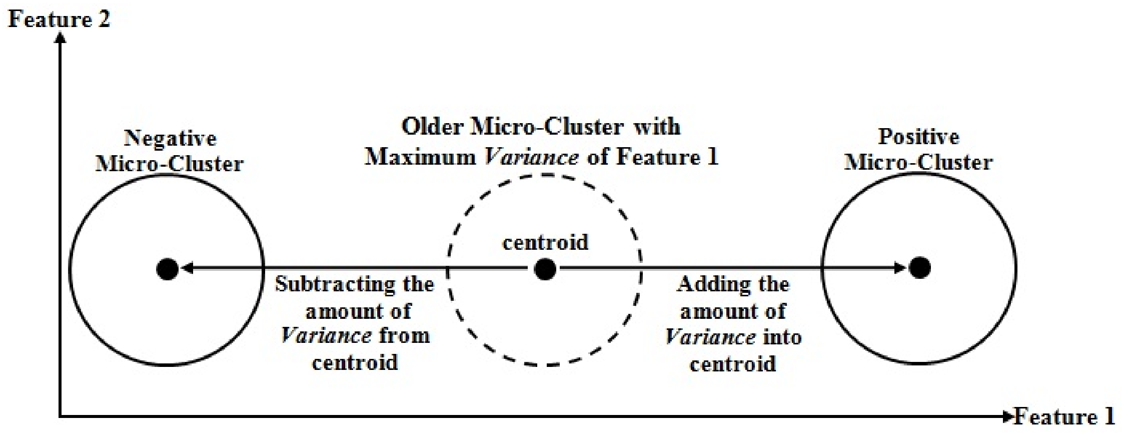

Feature 2
Negative Micro-Cluster
Older Micro-Cluster with Maximum *Variance* of Feature 1
Positive Micro-Cluster
centroid
Subtracting the amount of *Variance* from centroid
Adding the amount of *Variance* into centroid
Feature 1

Figure 3.5: Splitting of a Micro-Cluster according to the feature with the highest *Variance*.

The assumption made here for the research presented in this thesis is that continuing splitting of Micro-Clusters would indicate that a concept drift has happened and the Micro-Clusters do not fit as well anymore. Hence, this can be tracked through feeding forward statistical information over a time window for the purpose of concept drift detection, i.e., the *Split* rates over time. Whereas, the movement of the new Micro-Clusters which are generated from the original one can be tracked as well. This can be used for the purpose of identifying features which have contributed towards a concept drift.

3.4 Death and Removal of a Micro-Cluster using Triangle Numbers

MC-NN removes a Micro-Cluster if it has not participated recently in absorbing new data instances, which can be calculated from $CF1^t$ by measuring the *Triangle Number* (Equation 3.3). This is called Micro-Cluster *Death*. Figure 3.7 shows an example for calculating the *Triangle Number* of a Micro-Cluster.

$$\textit{Triangle Number}\ \Delta(T) = ((T^2+T)/2) \tag{3.3}$$

The *Triangle Number* gives more weight to recent Micro-Clusters than the older ones. If the participation percentage of a Micro-Cluster is lower than Ω, then the Micro-Cluster is removed. The assumption MC-NN makes is that older non-participating Micro-Clusters reflect old and potentially obsolete concepts. The assumption made here for the research presented in this thesis is that continuing removing of Micro-Clusters would indicate that a concept drift has happened and the Micro-Clusters do not fit as well anymore. Hence, this can be tracked through feeding forward statistical information over a time window for the purpose of concept drift detection, i.e., the *Death* rates over time.

The process of calculating the *Triangle Number* is given in Figure 3.6 and highlighted in a concrete example in Figure 3.7. Consider the Micro-Cluster in the example. It was created at time stamp 2 and updated with instances at time stamps 4 and 6, the current time stamp is 7 but here it was not updated. On the right-hand side of Figure 3.7 is depicted how the different steps in the process illustrated in Figure 3.6 are calculated.

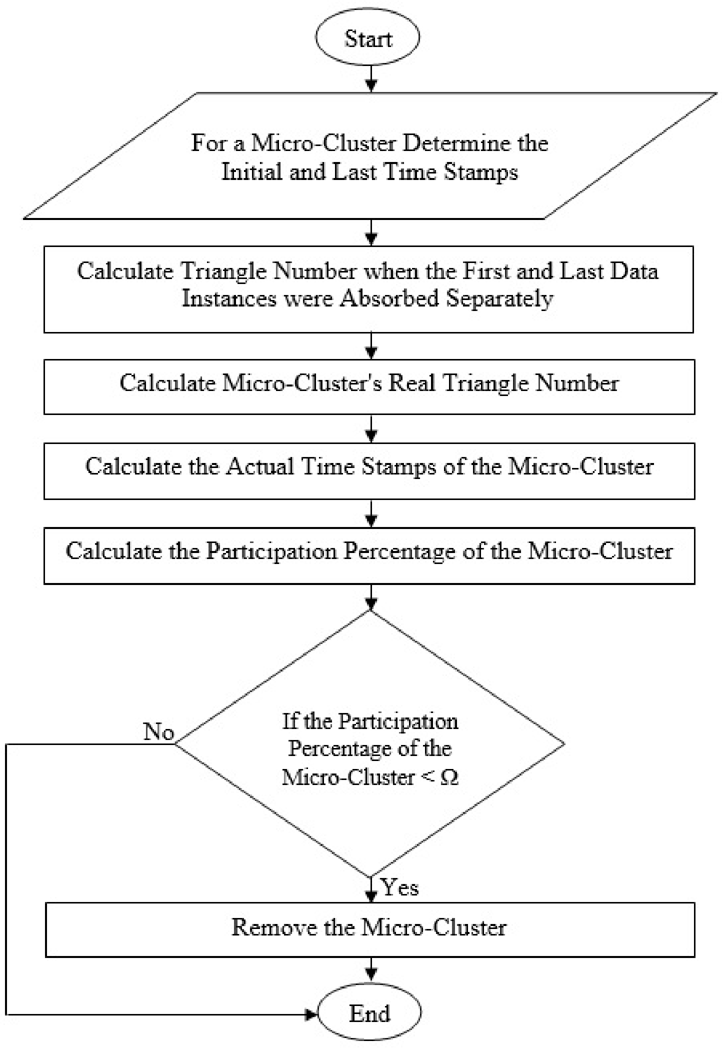

Start
For a Micro-Cluster Determine the Initial and Last Time Stamps
Calculate Triangle Number when the First and Last Data Instances were Absorbed Separately
Calculate Micro-Cluster's Real Triangle Number
Calculate the Actual Time Stamps of the Micro-Cluster
Calculate the Participation Percentage of the Micro-Cluster
If the Participation Percentage of the Micro-Cluster < Ω
No
Yes
Remove the Micro-Cluster
End

Figure 3.6: The process of calculating the *Triangle Number* for a Micro-Cluster.

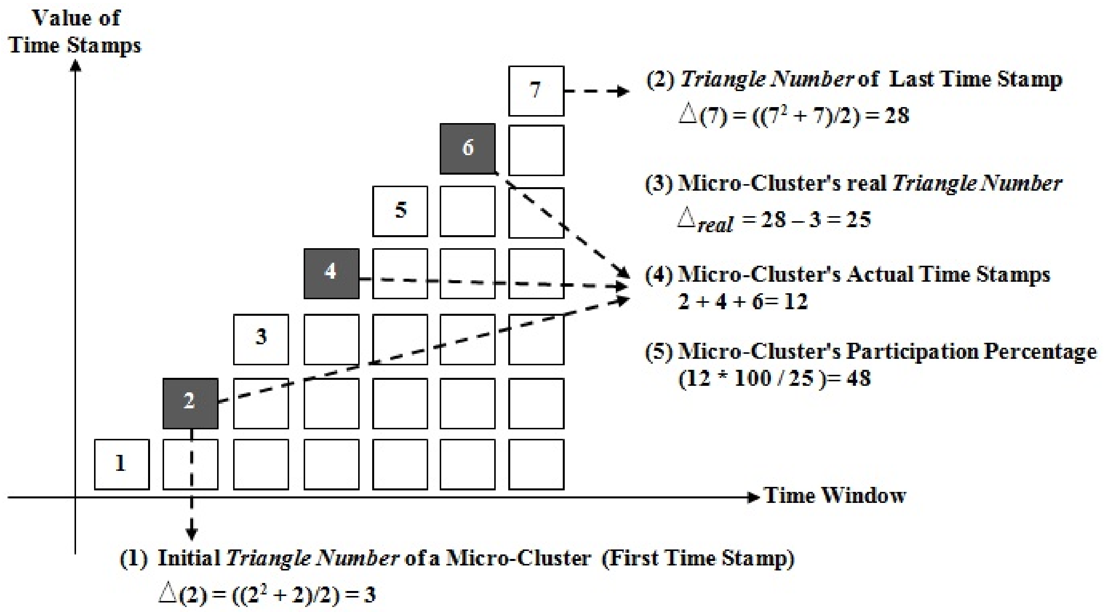

Value of Time Stamps
Time Window
(1) Initial Triangle Number of a Micro-Cluster (First Time Stamp)
△(2) = ((2² + 2)/2) = 3
(2) Triangle Number of Last Time Stamp
△(7) = ((7² + 7)/2) = 28
(3) Micro-Cluster's real Triangle Number
△real = 28 – 3 = 25
(4) Micro-Cluster's Actual Time Stamps
2 + 4 + 6= 12
(5) Micro-Cluster's Participation Percentage
(12 * 100 / 25)= 48

Figure 3.7: An example of *Triangle Number* calculation. The shaded areas signify the time stamps. Where the Micro-Cluster has participated in absorbing new instances for a specific time stamp.

3.5 Taking MC-NN Forward to Develop a Real-time Pre-Processing Technique

Chapters 4, 5 and 6 take some of MC-NN ideas forward to facilitate real-time pre-processing technique. The basic idea is to monitor Micro-Cluster *Split* and *Death* rates in order to detect concept drifts. It is expected that concept drifts will cause a peak in splitting and removing Micro-Clusters because they do not fit the concepts very well anymore. The *Velocity*, i.e. the rate and direction of movement of the Micro-Clusters can be used as an indication as to which features have changed and potentially contributed towards a concept drift. Then in turn features that have shown significant changes during a concept drift can be examined separately for their relevance to the classification task (i.e., real-time feature selection). A preliminary result which shows that the developed method is capable of detecting a concept drift but also delivers an indication which features are involved has been published in a conference paper (see Chapter 1).

For clarity in this research study note the semantic distinctions in the thesis usage of three terms relating to time and temporal referencing of a point or an interval in the timeline, namely ’time’, ’time window’, and ’time stamp’. In the thesis analysis ’time’ and ’time window’ are regarded as equivalent references used interchangeably to mean duration of time as encapsulating a set of data instances with sequential time intervals of fixed length. Whereas, ’time stamp’ refers to a specific data instance at a specific point in time. For example, assuming a stream has generated 2000 sequential data instances and each time window is of length 1000, then there would be two time windows, time window 1 from time stamp 0 to 999 and time window 2 from time stamp 1000 to 1999. This may be for example referred to in the thesis as time 1 or time 2 meaning the window 1 and 2.

3.6 Experimental Setup

The implementation of experiments was realised in the Massive Online Analysis (MOA) framework (Bifet et al., 2010). Two types of data were used, artificial data stream generators from the MOA framework and real datasets. The reason for using artificial datasets is because MOA’s data stream generators enable the introduction of different kinds of concept drift deliberately and thus allow this research to evaluate against a ground truth in terms of concept drift.

3.6.1 Artificial Datasets

The following artificial data stream generators were used:

SEA Generator, this data stream was introduced in work by (Street and Kim, 2001), it generates data comprising continuous attributes. Whereas, the third attribute is irrelevant for distinguishing between the class labels.

The ***HyperPlane Generator*** was also used, it creates a linearly separable model. It slowly rotates in ‘D’ dimensions continuously changing the linear decision boundary of the stream (Brzezinski and Stefanowski, 2014). This constant concept change makes it very difficult for data stream classifiers to keep a good classification accuracy and remain computationally efficient.

The final data stream generator used was the ***Random Tree Generator***, which was introduced in work reported by (Domingos and Hulten, 2000) and generates a stream based on a randomly generated tree. New examples are generated by assigning uniformly distributed random values to features, which then determine the class label using the random tree.

Fifteen datasets were generated using the aforementioned generators, each comprising three features, two class labels, and a concept drift. The concept drift was always induced halfway through the stream by both, inducing a *gradual concept drift* through the in MOA implemented data stream generators methods and by swapping features (swapping of features is considered as a *sudden concept drift*). The reason behind this is that at the beginning of *gradual concept drift* (before more data instances are seen), an instance might be mistaken as random noise. Therefore, a long period of time is required to detect the *gradual concept drift* (Žliobaitė, 2010; Brzezinski and Stefanowski, 2014). Hence, in the fifteen datasets, the *sudden concept drift* was generated after the *gradual concept drift* as known ground truth to show that the method presented in this research is robust to detect the actual/real concept drifts. However, other groups of artificial datasets with *gradual concept drift* and *recurring concept drift* induced as known ground truth through only swapping features are given in Appendices H and I, respectively, in order to show that the developed method is able to detect different types of concept drift. Details about the concept drift methods of the individual streams can be found in the works of (Domingos and Hulten, 2000; Street and Kim, 2001; Brzezinski and Stefanowski, 2014). The reason for inducing concept drift in this particular way is that this enables the testing as to which of the features has changed its contribution to the underlying model. One would expect the methods developed here to identify the swapped features as the cause of the concept drift. For each of these data stream generators, five datasets having concept drift, a second, third, fourth, and fifth version of the datasets has been generated which included different levels of noise in order to validate the robustness of the concept drift detection and feature tracking methods. Table 3.2 shows an overview of the generated streams including settings of the developed method and which features have been swapped. In order to increase the readability of the figures (i.e., results), in the experiments, a window size equals to 10% of the total number of instances. Hence, the expression **Time t** refers to a specific time window. I.e. according to the figures of the experiments in this chapter time *T=1* refers to instances 1-1000, *T=2* to instances 1001-2000, etc.

3.6.2 Real Datasets

Two sets of experiments were setup. One controlled set of experiments to validate whether the method can detect concept drifts and causality of concept drift on real datasets correctly. For this, features were swapped halfway through the stream to generate a known ground truth concept drift. Five real datasets with continuous features were chosen randomly from the UCI Machine Learning Repository (Lichman, 2013). Table 3.3 shows an overview of controlled real

Table 3.2: Setup of the artificial datasets. Drifts were generated through the individual data stream generators and by swapping features.

Dataset	Number of Instances Generated	Time of Concept Drift by Generator	Window Size	Θ	Index of Swapped Features	Time of Swapped Features	Percentage of Noise
SEA	10,000	5000	1000	3	2 with 3	6000	-
SEA	10,000	5000	1000	27	2 with 3	6000	5
SEA	10,000	5000	1000	34	2 with 3	6000	15
SEA	10,000	5000	1000	75	2 with 3	6000	25
SEA	10,000	5000	1000	155	2 with 3	6000	35
HyperPlane	10,000	5000	1000	6	1 with 2	6000	-
HyperPlane	10,000	5000	1000	6	1 with 2	6000	5
HyperPlane	10,000	5000	1000	13	1 with 2	6000	15
HyperPlane	10,000	5000	1000	27	1 with 2	6000	25
HyperPlane	10,000	5000	1000	59	1 with 2	6000	35
Random Tree	10,000	5000	1000	644	1 with 2	6000	-
Random Tree	10,000	5000	1000	660	1 with 2	6000	5
Random Tree	10,000	5000	1000	726	1 with 2	6000	15
Random Tree	10,000	5000	1000	685	1 with 2	6000	25
Random Tree	10,000	5000	1000	200	1 with 2	6000	35

datasets including settings of the developed method and which features were swapped. Two versions of each dataset were used with two features (indexes of swapped features are 1 with 4), and four features (indexes of swapped features are 2 and 3 with 5 and 6) have been selected randomly to be swapped in order to validate whether the feature tracking method can identify the changed features correctly.

Table 3.3: Setup of the real datasets for the controlled set of experiments for concept drift detection and feature tracking.

Dataset	Number of Instances	Number of Features	Number of Class Labels	Θ	Index of Randomly Swapped Features	Time of Swapping
CoverType	581,012	6	7	6,000	1 with 4	6
CoverType	581,012	6	7	35,500	2 and 3 with 5 and 6	6
Diabetic Retinopathy Debrecen	1,151	6	2	95	1 with 4	6
Diabetic Retinopathy Debrecen	1,151	6	2	57	2 and 3 with 5 and 6	6
Gesture Phase Segmentation	1,747	6	5	73	1 with 4	6
Gesture Phase Segmentation	1,747	6	5	74	2 and 3 with 5 and 6	6
Statlog (Landsat Satellite)	4,435	6	7	50	1 with 4	6
Statlog (Landsat Satellite)	4,435	6	7	251	2 and 3 with 5 and 6	6
Waveform (with Noise)	5,000	6	3	301	1 with 4	6
Waveform (with Noise)	5,000	6	3	360	2 and 3 with 5 and 6	6

The second set of experiments was uncontrolled, five real datasets were used, and no features were swapped, in order to show that the method presented in this research is robust to more realistic application scenarios. It should be noted that the controlled real datasets mentioned above (Table 3.3) have been re-used for the experiments here (real-time pre-processing technique) under uncontrolled conditions. This time all features were included. It is expected that these original versions of real datasets are more challenging for data stream algorithms to induce good models compared with reduced versions described in Table 3.3. The developed real-time pre-processing technique which consists of concept drift detection method, feature tracking method and also real-time feature selection method was applied. Table 3.4 shows an overview of uncontrolled real datasets including settings of the developed technique. The robustness of the technique was measured by applying a Hoeffding tree classifier, and the classification accuracy was monitored over time windows. Also, a control group of experiments was setup. Where only a Hoeffding tree classifier was applied thus excluding any of the methods developed in this research work. The Hoeffding tree classifier has been chosen as it is one of the most popular data stream classifiers and best performing classifier in the MOA framework (Marrón et al., 2017). However, the developed technique is independent of the classifier used, and the user may choose a different classification method.

Table 3.4: Setup of real datasets for the uncontrolled set of experiments for concept drift detection, feature tracking and real-time feature selection.

Real Dataset	Number of Instances	Number of Features	Number of Class Labels	*FIFO*'s Size (Maximum)	Θ
CoverType	581,012	54	7	1000	50,000
Diabetic Retinopathy Debrecen	1,151	19	2	1000	6
Gesture Phase Segmentation	1,747	19	5	1000	56
Statlog (Landsat Satellite)	4,435	36	7	1000	28
Waveform (with noise)	5,000	40	3	1000	175

In order to increase the readability of the figures (i.e., results), for each real dataset in the experiments, a window size was set to a value that is either less or equal to 10% of the number of instances. In the figures, the expression **Time t** refers to a specific time window. I.e. according to figures (results) time $T{=}1$ refers to instances 1-500, $T{=}2$ to instances 501-1000, etc. The classification accuracy was calculated using the Prequential Testing method implemented in MOA (Bifet and Frank, 2010), which essentially calculates a running average of the classification accuracy.

For the experiments in this chapter, Ω (i.e., a minimum (user-defined) threshold for the Micro-Cluster minimum participation) was set to 50 as this yielded good results in most cases, as stated in (Tennant et al., 2017). The *Percentage Difference* of *Split* and *Death* rates and α rate of Low Pass Filter (*LPF*) were set to 50% as it yielded good results for each individual dataset as examined in Appendix A. Whereas, Θ was set to a relevant value that yielded good results for each individual dataset as examined in Appendix D. Regarding the *FIFO* queue, it was set to a 1000 instances, as this setting yielded good results in most cases as examined in Appendix B.

3.7 Summary

This chapter introduced MC-NN algorithm which aims to keep a recent accurate summary of the data stream using Micro-Clusters. The statistical summaries such as $Split$ and $Death$ rates can be calculated using Micro-Clusters for the purpose of drift detection. This research study is giving focus more on MC-NN rather than the aforementioned methods and algorithms presented in Chapter 2 because of these useful properties of MC-NN. This will be introduced in the next chapter.

Chapter 4

Real-Time Concept Drift Detection Method using Adaptive Micro-Clusters

The work in this chapter directly links to Objective 1 to identify a drift point (i.e., concept drift) through tracking the significant changes in the statistical summaries in real-time. The work described in this chapter is based on the Micro-Cluster structure of the MC-NN algorithm presented in Chapter 3. The MC-NN algorithm aims to keep a recent and accurate summary of the data stream using Micro-Clusters. Significant changes to these summaries (i.e., historical statistics) are used in this research to detect concept drift. The chapter is organised as follows. In Section 4.1, detecting concept drift using adaptive Micro-Clusters is introduced. Section 4.2 presents a working example which shows how a concept drift is detected. The performance, implementation, and results of the developed method are discussed in Section 4.3 with respect to real and artificial datasets. In Section 4.4, a summary of this chapter is presented.

4.1 Detecting Concept Drift using Adaptive Micro-Clusters

It is expected that during a concept drift the data distribution of Micro-Clusters changes (causing larger *Variance* within Micro-Clusters), but also it is expected that Micro-Clusters absorb more incorrect data instances (with different class labels than the Micro-Cluster), as explained in Chapter 3. MC-NN consists of two main tasks which are the splitting and removal of Micro-Clusters (referred to as *Split* and *Death*). When new concepts arrive, MC-NN adapts by absorbing and applying *Split* and *Death*, as explained in Chapter 3. Loosely speaking, regarding *Split*, two new Micro-Clusters are generated from the original one when the error count ε reaches Θ

threshold due to false positive participation. Likewise, some of the existing Micro-Clusters may stop to participate as they become obsolete and are removed. In this research this is measured as *Split* and *Death* rates over time simply by using Equation 4.1 for each time window. Here *n* is the number of instances in a time window and *i* is the index of the current data instance within the current time window, starting with 1 within each time window. Figure 4.1 illustrates an example of monitoring a *Split* or *Death* rate. In this sense, the calculation of *Split* and *Death* is considered gradually instance by instance within a time window. Higher rates of *Splits* and *Deaths* are expected over time. This would indicate that new concepts have arrived or have changed (see Chapter 3). Hence, a peak in *Split* and *Death* rates is used as a trigger to detect the concept drift.

$$Split\ or\ Death\ rate = \frac{\sum_{i=1}^{n}(Number\ of\ Splits\ or\ Deaths_i - Number\ of\ Splits\ or\ Deaths_{i-1})}{n} \tag{4.1}$$

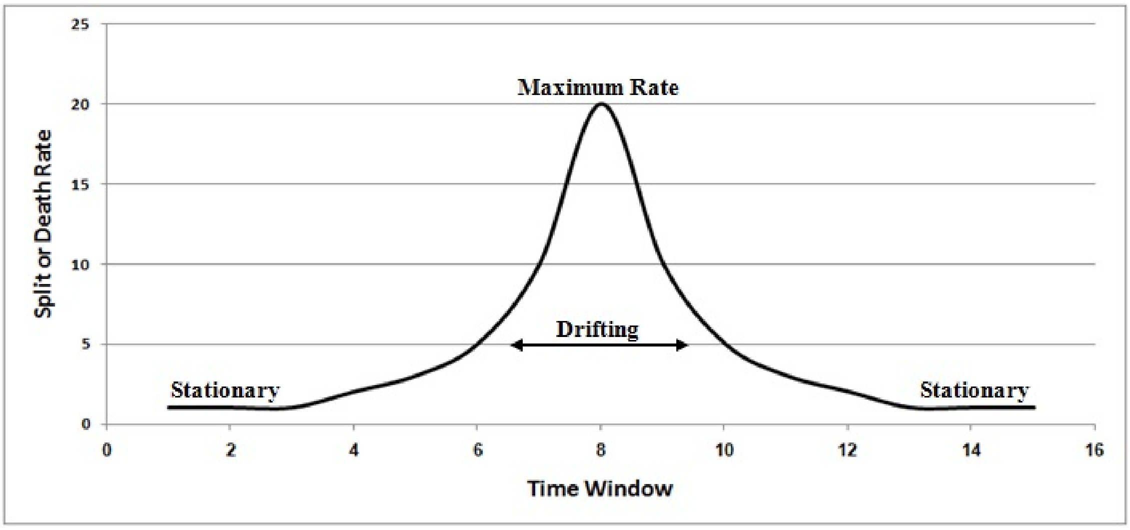


Figure 4.1: An example of Micro-Cluster *Split* and *Death* rate.

The assumption is that the larger both rates are, the more likely it is that a concept drift has happened. Using a windowing approach on the data stream, running averages of the *Split* and *Death* rates are calculated, as well as the *Percentage Difference* in comparison with the previous time window which is given by Equation 4.2. In the equation *i* denotes the current time window.

$$Percentage\ Difference = \frac{|\ Rate_i - Rate_{i-1}\ |}{(Rate_i + Rate_{i-1})/2} * 100 \tag{4.2}$$

If the *Percentage Differences* of both, the *Split* and *Death* rates differ from the μ value (of the current and previous window) by 50% (default value), then this is considered a concept drift. In this work, the default value of 50% has been used for all experiments as it yielded good results in most cases as examined in Appendix A. However, the user may change this to a different threshold. Keeping old concepts would potentially render a concept drift detection method unable to detect unexpected changes in data streams (Wang et al., 2013). Hence, once a concept drift is detected, all Micro-Clusters are re-initialised. This is illustrated in Algorithm 1.

Algorithm 1 Detection of Drifts

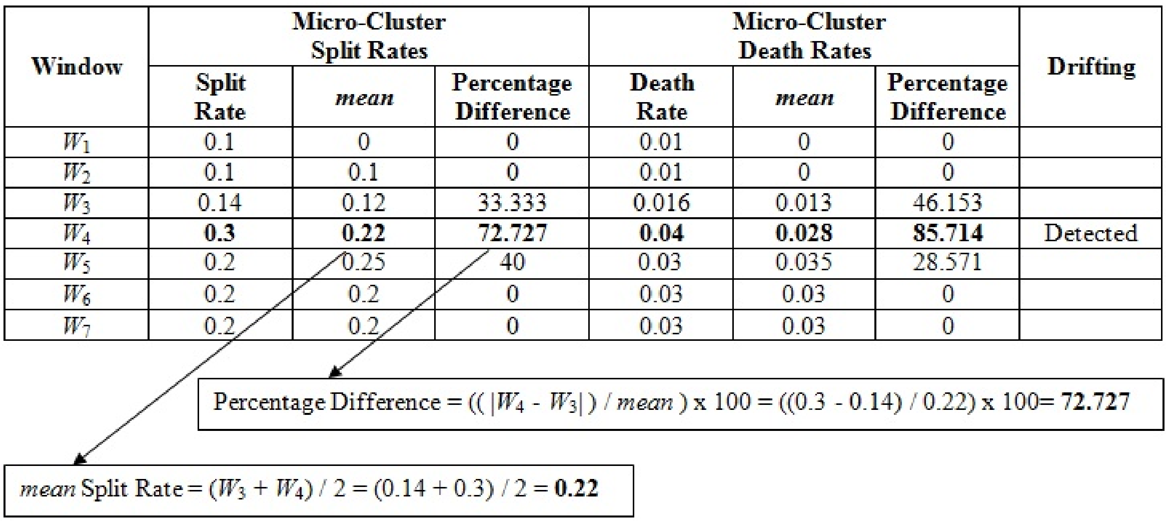
Input: Micro-Cluster(s) *Split* and *Death* rates for current and previous window
Output: Detection of Drifts

```
1: Calculate the $\mu$ of Split and Death rates over the current and previous window
2: Calculate the Percentage Difference for both, Split and Death rates
3: if new Split rate $>\mu$ (Split rate) AND new Death rate $>\mu$ (Death rate) AND Percentage
   Difference (Split rate) $>50\%$ AND Percentage Difference (Death rate) $>50\%$ then
4:     Concept Drift
5:     re-initialise MC(s)
6: else
7:     No Concept Drift
8: end if
```

A working example is presented in Section 4.2 below.

4.2 Worked Example

Split and *Death* rates of Micro-Clusters are as shown below in Figure 4.2 for seven consequently time windows.



Window	Micro-Cluster Split Rates			Micro-Cluster Death Rates			Drifting
	Split Rate	*mean*	Percentage Difference	Death Rate	*mean*	Percentage Difference	
W_1	0.1	0	0	0.01	0	0	
W_2	0.1	0.1	0	0.01	0	0	
W_3	0.14	0.12	33.333	0.016	0.013	46.153	
W_4	**0.3**	**0.22**	**72.727**	**0.04**	**0.028**	**85.714**	Detected
W_5	0.2	0.25	40	0.03	0.035	28.571	
W_6	0.2	0.2	0	0.03	0.03	0	
W_7	0.2	0.2	0	0.03	0.03	0	

Percentage Difference = (($|W_4 - W_3|$) / *mean*) x 100 = ((0.3 - 0.14) / 0.22) x 100= **72.727**

mean Split Rate = ($W_3 + W_4$) / 2 = (0.14 + 0.3) / 2 = **0.22**

Figure 4.2: An example of concept drift detection using the Micro-Clusters *Split* and *Death* rates.

Consider the *Split* and *Death* rates of window (W_4), which are equal to 0.3 and 0.04, separately. This shows that the *Split* and *Death* rates differ from the μ values (i.e., 0.22 and 0.028) by more than 50% as the *Percentage Difference* of *Split* rate equals to 72.727, and the Percentage Difference of *Death* rate equals to 85.714. This indicates that a concept drift has occurred. Next, the causality of concept drift is tracked through feeding forward the statistical information such as *Velocity* of the features. This will be described in Chapter 5. The performance, implementation, and results of the developed method are discussed and recorded in Section 4.3 with respect to different real and artificial datasets.

4.3 Empirical Evaluation of Real-Time Concept Drift Detection Method

The Micro-Clusters *Split* and *Death* rates were used for detecting drifts. For the experiment the default parameters stated in Table 3.2 of the method were used unless stated otherwise. The evaluation incorporated several levels of noise in the artificial data stream; the different levels of noise were introduced as listed in Table 3.2 and the real datasets were described in Table 3.3.

In Figures 4.3 to 4.5 the *Percentage Difference* is displayed up to 100%, however this can be much higher than 100%. In order to increase the readability of the figures (i.e., results), a maximum of 100% difference is displayed. Displaying differences above 100% is not very interesting as the concept drift detection is triggered once a difference of at least 50% was reached for both *Split* and *Death* rates. This 100% cut-off is also applied on all subsequent figures in this research referring to *Percentage Differences* of *Split* and *Death* rates. During the time of concept drift higher *Split* and *Death* rates are expected as the set of Micro-Clusters adapts to the new concept and the feature swap. In the figures, it can be seen that the *Split* and *Death* percentage differences at the time of concept drift (after time 5) increase considerably as expected for all artificial data streams and noise levels. The noise levels do not seem to affect the concept drift detection considerably; in two cases, however, for a noise level of 25% and 35%, the algorithm did arrive at some *false positive* detections, i.e., detected concept drifts that were not there (i.e., unknown ground truth). This could be an indication that the Micro-Clusters for *SEA* generator, *HyperPlane* generator, and *RandomTree* generator became unstable due to noise. This will be discussed further in Section 5.4 where mechanisms to deal with noise are incorporated in the feature tracking method. The actual values of *Split* and *Death* rates for the

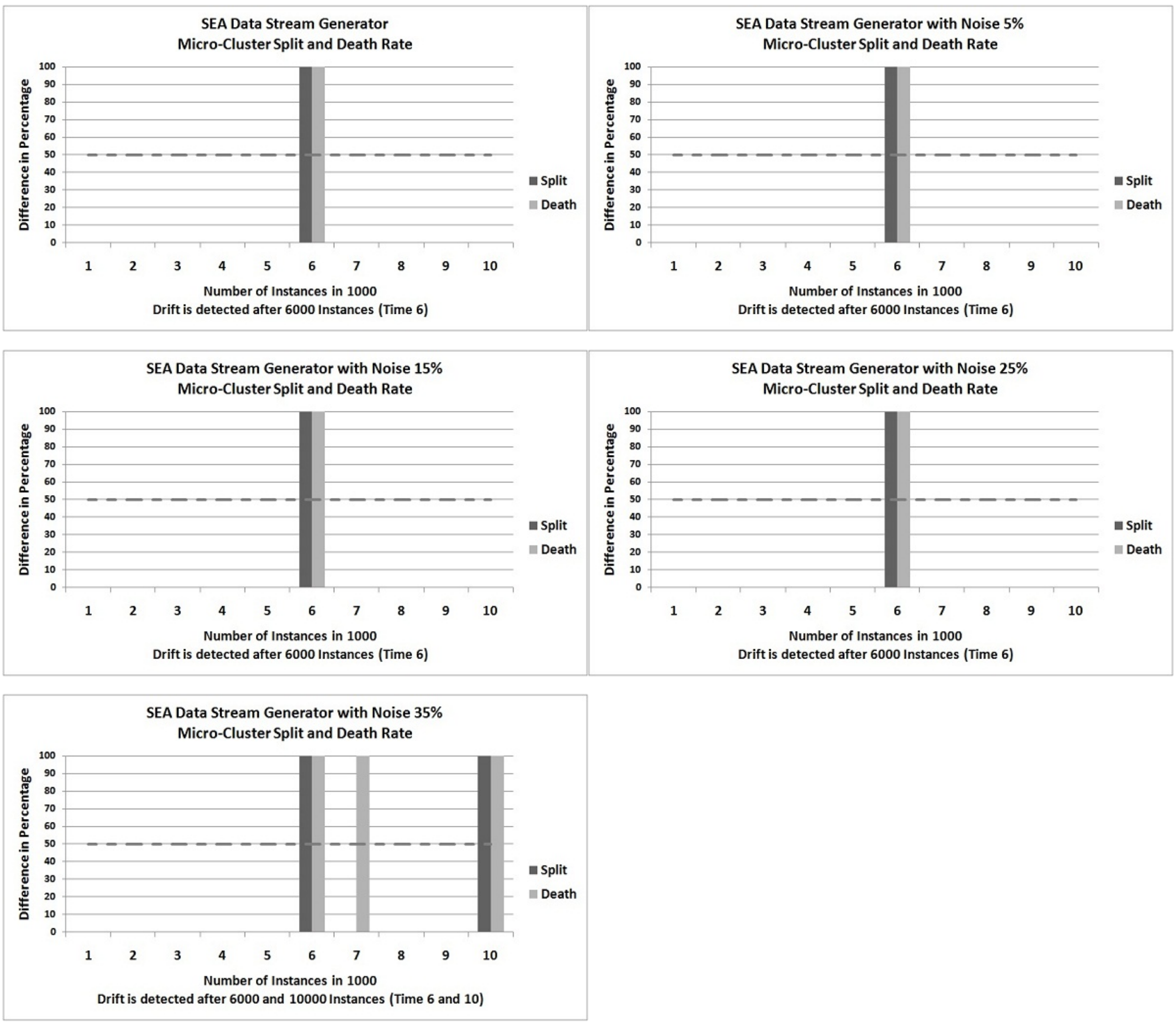


Figure 4.3: The results of *SEA* Data Stream Generator using Micro-Cluster *Percentage Difference* of *Split* and *Death* rates for drift detection.

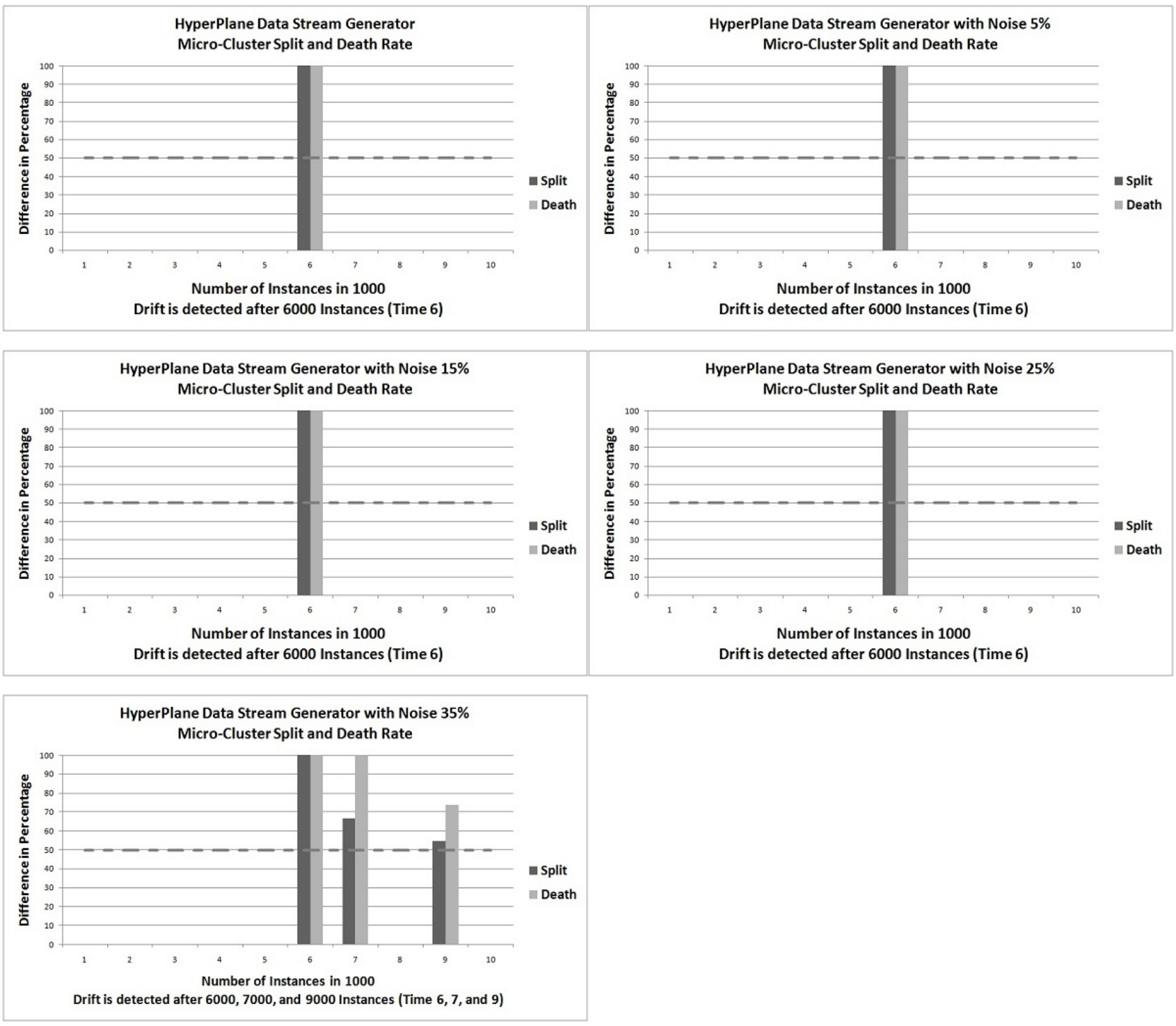


Figure 4.4: The results of *HyperPlane* Data Stream Generator using Micro-Cluster *Percentage Difference* of *Split* and *Death* rates for drift detection.

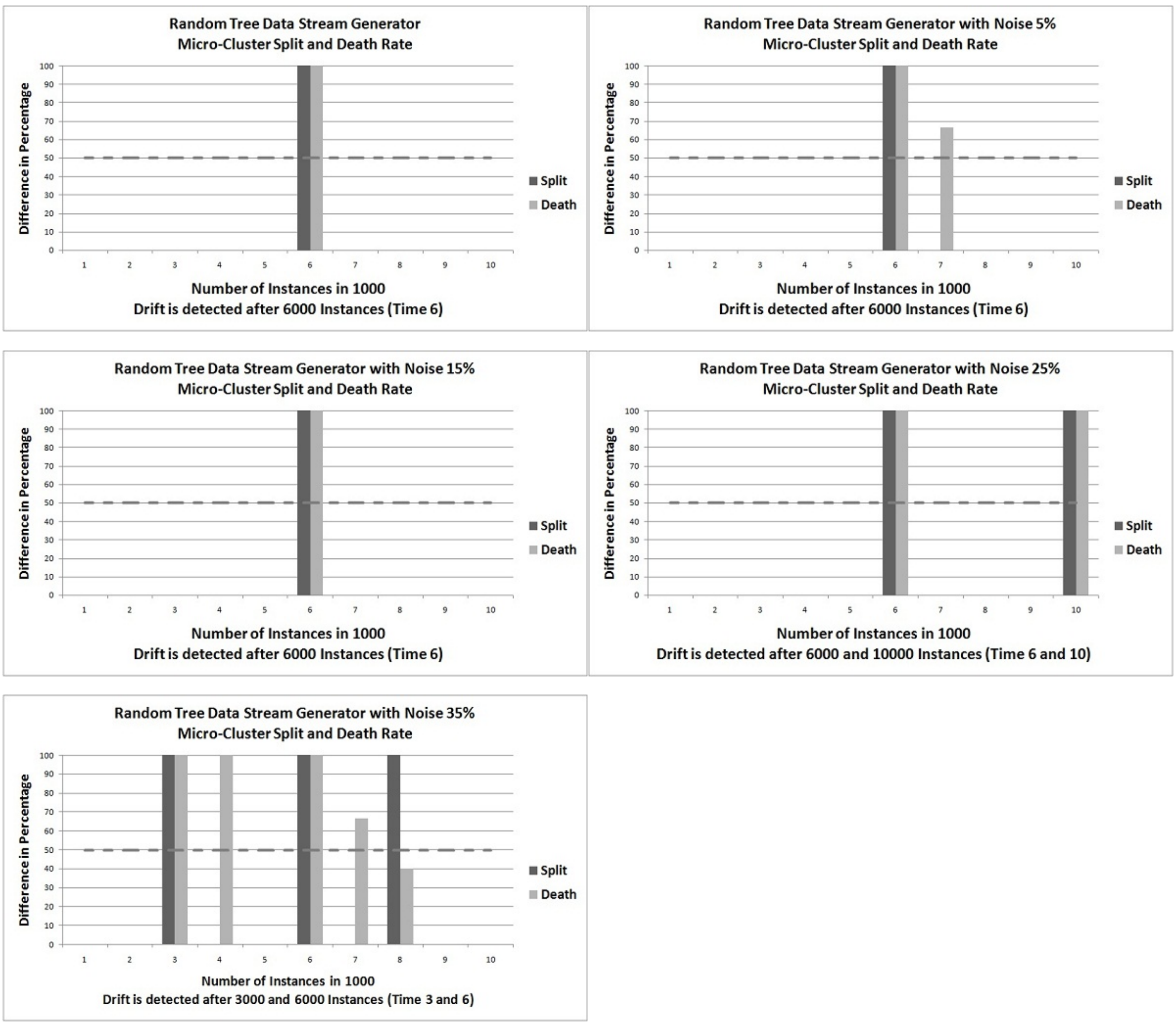


Figure 4.5: The results of *Random Tree* Data Stream Generator using Micro-Cluster *Percentage Difference* of *Split* and *Death* rates for drift detection.

results shown in Figures 4.3 to 4.5 are given in the Appendix F.

Next, the concept drift detection method was compared with existing state-of-the-art drift detection methods CUSUM, DDM, EDDM, EWMA, and ADWIN (see Chapter 2 for more details about these methods) on the same artificial data streams. They are considered the most well-known drift detection methods that are frequently used (Sidhu and Bhatia, 2015; De Barros et al., 2018). In addition, they are found in MOA (Bifet et al., 2010). Table 4.1 shows the time when each of the methods including the developed method, detected a drift and if it was detected on time. As it can be seen, the developed method always detected the drift at the correct time except for the *Random Tree* data stream with 25% and 35% noise. This could be because of Micro-Clusters becoming unstable due to noise.

If a method detects a drift falsely, then this would cause unnecessary adaptation by the classifier to a non-existing drift. This is referred to as a *false positive*. Correctly detected drifts are referred to as *true positives*. The outcome of the experiments (each with one actual drift) is shown in Table 4.2. As there are 15 experiments, there is a total of 15 concept drifts to be detected. In the table it is indicated how many *true* and *false positives* each method detected. As can be seen, the developed method detected all concept drifts and only had 5 *false positives* detection. The best competitor in this regard, EWMA, it detected 10 *true positives*, 5 less than the developed method. However, EWMA has a very high *false positives* number (113), compared with only 5 for the developed method. Thus EWMA is triggering frequent and unnecessary adaptation to concept drift. Also, the remaining competitors found fewer *true positives* and a much higher number of *false positives* compared with the developed method.

The method has also been applied on real datasets as shown in Figures 4.6 to 4.10 where for each case 1 concept drift has been introduced through the swapping of features as listed in Table 3.3.

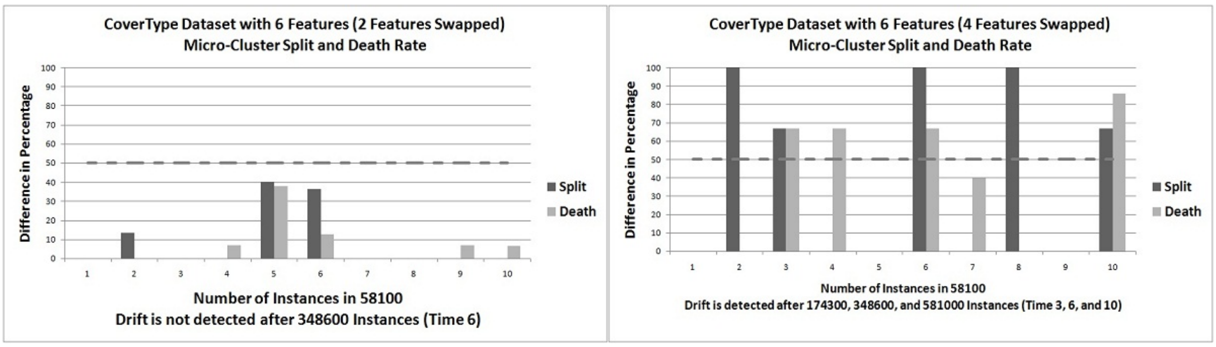


Figure 4.6: The results of *CoverType* Dataset with 6 Features using Micro-Cluster *Percentage Difference* of *Split* and *Death* rates for drift detection.

Table 4.1: Adaptation to concept drift using the initially developed and other state-of-the-art methods.

Generator	Method	Number of Drift Detections	Times when Drift Detected	Drift Detected
SEA	**The Developed Method**	1	6	**Correctly**
	CUSUM	1	5	Incorrectly
	DDM	2	5 and 8	Incorrectly
	EDDM	2	5 and 8	Incorrectly
	EWMA	9	1 to 9	**Correctly**
	ADWIN	1	5	Incorrectly
SEA with Noise 5	**The Developed Method**	1	6	**Correctly**
	CUSUM	1	5	Incorrectly
	DDM	1	5	Incorrectly
	EDDM	1	5	Incorrectly
	EWMA	6	2 to 5, 7, and 8	Incorrectly
	ADWIN	3	1, 5, and 6	**Correctly**
SEA with Noise 15	**The Developed Method**	1	6	**Correctly**
	CUSUM	2	5 and 8	Incorrectly
	DDM	1	5	Incorrectly
	EDDM	2	1 and 4	Incorrectly
	EWMA	9	1 to 9	**Correctly**
	ADWIN	2	5 and 9	Incorrectly
SEA with Noise 25	**The Developed Method**	1	6	**Correctly**
	CUSUM	2	5 and 6	**Correctly**
	DDM	1	4	Incorrectly
	EDDM	3	1,4, and 6	**Correctly**
	EWMA	9	1 to 9	**Correctly**
	ADWIN	3	4,5, and 8	Incorrectly
SEA with Noise 35	**The Developed Method**	2	6 and 10	**Correctly**
	CUSUM	2	4 and 8	Incorrectly
	DDM	2	4 and 6	**Correctly**
	EDDM	2	4 and 5	Incorrectly
	EWMA	6	2 to 5,7, and 8	Incorrectly
	ADWIN	4	1,4,5, and 9	Incorrectly
HyperPlane	**The Developed Method**	1	6	**Correctly**
	CUSUM	1	5	Incorrectly
	DDM	1	5	Incorrectly
	EDDM	1	5	Incorrectly
	EWMA	8	1 to 5 and 7 to 9	Incorrectly
	ADWIN	3	2,5, and 9	Incorrectly
HyperPlane with Noise 5	**The Developed Method**	1	6	**Correctly**
	CUSUM	1	5	Incorrectly
	DDM	1	5	Incorrectly
	EDDM	1	5	Incorrectly
	EWMA	7	1,2,4 to 6,8, and 9	**Correctly**
	ADWIN	2	2 and 5	Incorrectly
HyperPlane with Noise 15	**The Developed Method**	1	6	**Correctly**
	CUSUM	1	5	Incorrectly
	DDM	2	5 and 7	Incorrectly
	EDDM	1	5	Incorrectly
	EWMA	9	1 to 9	**Correctly**
	ADWIN	2	2 and 5	Incorrectly
HyperPlane with Noise 25	**The Developed Method**	1	6	**Correctly**
	CUSUM	2	5 and 6	**Correctly**
	DDM	2	5 and 6	**Correctly**
	EDDM	2	4 and 7	Incorrectly
	EWMA	9	1 to 9	**Correctly**
	ADWIN	3	2,5, and 8	Incorrectly
HyperPlane with Noise 35	**The Developed Method**	3	6,7,and 9	**Correctly**
	CUSUM	2	5 and 6	**Correctly**
	DDM	2	4 and 7	Incorrectly
	EDDM	2	4 and 6	**Correctly**
	EWMA	9	1 to 9	**Correctly**
	ADWIN	4	2 and 4 to 6	**Correctly**
Random Tree	**The Developed Method**	1	6	**Correctly**
	CUSUM	1	5	Incorrectly
	DDM	1	5	Incorrectly
	EDDM	1	5	Incorrectly
	EWMA	9	1 to 9	**Correctly**
	ADWIN	2	5 and 9	Incorrectly
Random Tree with Noise 5	**The Developed Method**	1	6	**Correctly**
	CUSUM	-	-	-
	DDM	-	-	-
	EDDM	-	-	-
	EWMA	7	1 to 5,7, and 8	Incorrectly
	ADWIN	2	5 and 9	Incorrectly
Random Tree with Noise 15	**The Developed Method**	1	6	**Correctly**
	CUSUM	1	5	Incorrectly
	DDM	1	5	Incorrectly
	EDDM	1	5	Incorrectly
	EWMA	9	1 to 9	**Correctly**
	ADWIN	2	5 and 9	Incorrectly
Random Tree with Noise 25	**The Developed Method**	2	6 and 10	**Correctly**
	CUSUM	1	7	Incorrectly
	DDM	-	-	-
	EDDM	-	-	-
	EWMA	9	1 to 9	**Correctly**
	ADWIN	-	-	-
Random Tree with Noise 35	**The Developed Method**	2	3 and 6	**Correctly**
	CUSUM	1	5	Incorrectly
	DDM	1	5	Incorrectly
	EDDM	2	5 and 7	Incorrectly
	EWMA	8	1 to 5 and 7 to 9	Incorrectly
	ADWIN	2	5 and 7	Incorrectly

Table 4.2: Summary of concept drift adaptation experiments referring to Table 4.1.

Method	True Positives	False Positives
The Developed Method	**15**	**5**
CUSUM	3	16
DDM	2	16
EDDM	2	19
EWMA	10	113
ADWIN	2	33

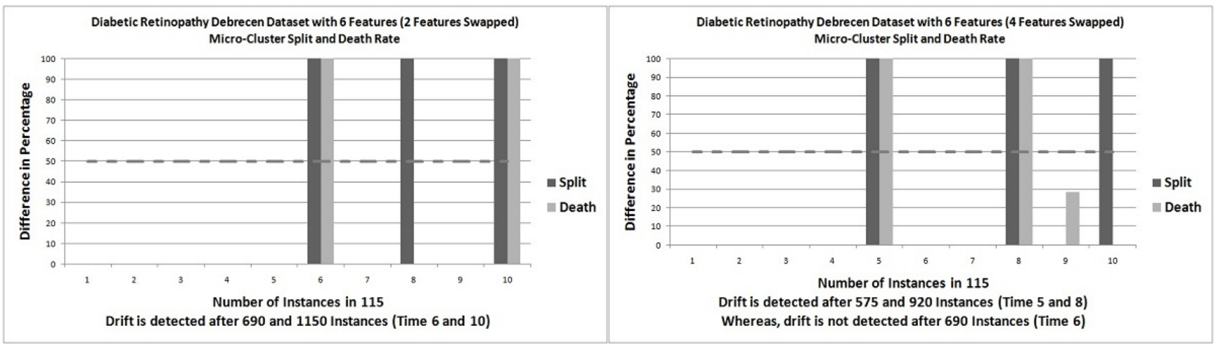


Figure 4.7: The results of *Diabetic Retinopathy Debrecen* Dataset with 6 Features using Micro-Cluster *Percentage Difference* of *Split* and *Death* rates for drift detection.

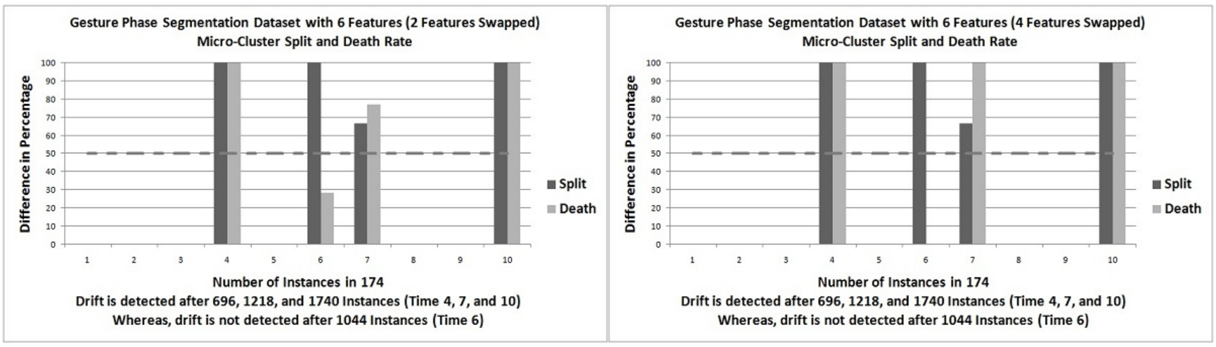


Figure 4.8: The results of *Gesture Phase Segmentation* Dataset with 6 Features using Micro-Cluster *Percentage Difference* of *Split* and *Death* rates for drift detection.

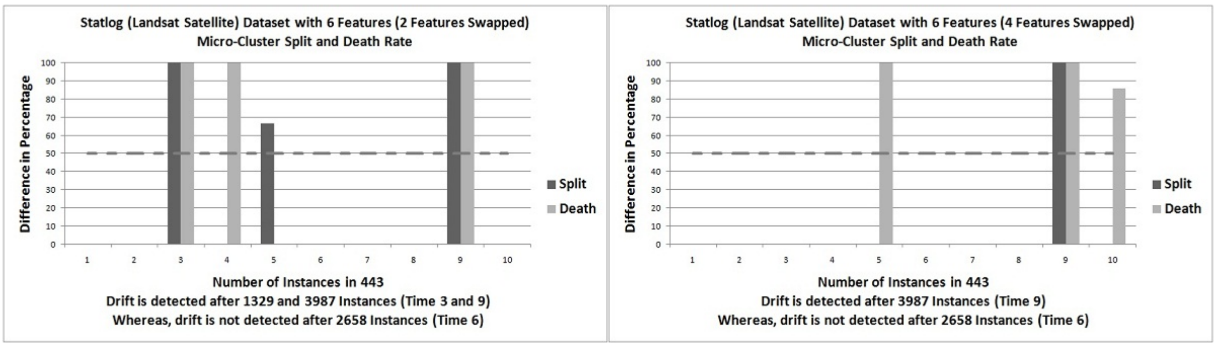


Figure 4.9: The results of *Statlog (Landsat Satellite)* Dataset with 6 Features using Micro-Cluster *Percentage Difference* of *Split* and *Death* rates for drift detection.

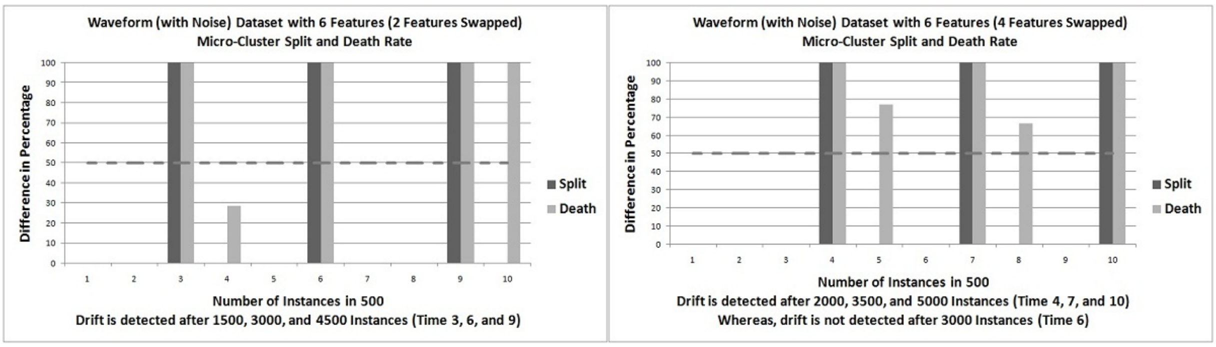


Figure 4.10: The results of *Waveform (with Noise)* Dataset with 6 Features using Micro-Cluster *Percentage Difference* of *Split* and *Death* rates for drift detection.

In Figures 4.6, 4.7, and 4.10, the *CoverType* dataset (with 4 features swapped), the *Diabetic Retinopathy Debrecen* dataset (with 2 features swapped), and the *Waveform* dataset (with 2 features swapped), it can be seen that concept drift is detected correctly as the $Split$ and *Death* percentage differences at the time of concept drift increase, indicating that the current set of Micro-Clusters does not fit the concept encoded in the data anymore. Whereas, in Figures 4.6 to 4.10, the *CoverType* dataset (with 2 features swapped), the *Diabetic Retinopathy Debrecen* dataset (with 4 features swapped), the *Gesture Phase Segmentation* dataset, the *Statlog (Landsat Satellite)* dataset, and the *Waveform* dataset (with 4 features swapped), drift at time of swapped features (i.e., known ground truth) is not detected. As these data streams are based on real datasets, it is believed that they potentially contain feature-bias, noise, and outliers and that these may be the reasons for the developed method not being able to detect the concept drift. The next chapter will compare this method with the newly developed concept drift detection method in combination with the new feature tracking method, which is expected to be more robust to feature-bias, noise, and outliers. The actual values of *Split* and *Death* rates for the results shown in Figures 4.6 to 4.10 are given in Appendix F.

4.4 Conclusion

This chapter introduced a novel Micro-Cluster based method for drift detection in data streams. The MC-NN algorithm presented in Chapter 3 aims to keep a recent accurate summary of the data stream using Micro-Clusters. The statistical summaries such as *Split* and *Death* rates are calculated using Micro-Clusters for the purpose of drift detection. The analyses show that the method did detect concepts drifts very well on the artificial data streams compared with alternative concept drift detection methods. On the controlled real datasets seven artificially induced concept drifts were missed, but unknown concept drifts were detected. As these data streams are based on real datasets, it is believed that they potentially contain feature-bias, noise, and outliers and that these may be the reasons for the developed method not being able to detect the concept drift. The next chapter will compare this method with the newly developed concept drift detection method in combination with the new feature tracking method, which is expected to be more robust to feature-bias, noise, and outliers. The next chapter also highlights feature tracking, which is invoked after a concept drift is detected to examine the causality of the drift for feature selection purposes.

Chapter 5

Real-Time Feature Tracking Method using Adaptive Micro-Clusters

The work in this chapter directly links to Objective 2 to detect the causality of drifts (which is identified by the developed method in Objective 1, i.e., Chapter 4) through providing the historical statistics of each feature for identifying which features were involved in drifting over a statistical time window in real-time. This chapter describes the developed method for tracking features that have been involved in a concept drift by monitoring statistical information such as the change of *Velocity* and *Variance* of features. These properties are derived from MC-NN Micro-Clusters and are used once a concept drift is detected as described in Chapter 4. *Velocity* and *Variance* are calculated for each feature of a Micro-Cluster over time stamps. The *Velocity* and *Variance* of the features are then analysed to identify features that have been involved in the drift. This information can be used for feature selection purposes which will be explained in Chapter 6. Feature-bias, outliers, and noise potentially influence the tracking of features. Thus the here presented method incorporates Min-Max *Normalisation*, *IQR*, and *LPF* to counter these influences. The remainder of this chapter discusses in Section 5.1 how the proposed method addresses potential feature bias, in Section 5.2 it is explained how the method addresses the problem of noise, Section 5.3 then explains how the features are tracked and outliers are taken into consideration. The performance, implementation, and results of the developed method are discussed in Section 5.4 with respect to real and artificial datasets. Section 5.5 summarises this chapter.

5.1 Minimising the Effect of Feature-Bias using Real-Time Min-Max Normalisation

In this research, a Min-Max *Normalisation* technique was used as minimum and maximum values of a feature *x* can easily be re-initialised in real-time when new instances arrive. Equation 2.1 (see Chapter 2) is applied for every new data instance as shown in Figure 5.1 which shows a flowchart of Min-Max *Normalisation*. The alternative techniques explained in Chapter 2 (Decimal Scaling and Z-Score) rely on the σ and the μ and thus require the buffering of data before *Normalisation* can be applied. This is undesirable in real-time data analytics. However, the *Normalisation* process used in this research study, Min-Max *Normalisation*, can be updated incrementally instance by instance. A pre-defined boundary such as [0,100] was used. Old instances cannot re-normalised as the original data values are not buffered but absorbed in the statistics of a Micro-Cluster. This could lead to Micro-Clusters not fitting the current concept well anymore. However, this would also contribute to the detection of the concept drift (Micro-Cluster *Split*) as the *Variance* of the Micro-Cluster would increase and potentially also the error count ε. Figure 5.2 shows an example of Min-Max *Normalisation* in real-time.

Next, the Micro-Cluster by which the normalised data instance should be absorbed has to be determined. For this, the absorbing task of MC-NN described in Chapter 3 is applied. However, before the data instance is absorbed, a Low Pass Filter (*LPF*) is applied to the nearest Micro-Cluster to minimise the effect of noise. This is explained in the next section.

5.2 Minimising the Effect of Noise using Low Pass Filter (LPF)

In this research study, *LPF* is used as it can be calculated within a time window (i.e., over time stamps) without buffering of data comparing to alternative techniques discussed in Chapter 2 (Kalman filter and Grid-based filter) which require buffering of data before filtering (Arulampalam et al., 2002). Each normalised feature of a new training instance was filtered by *LPF* together with its nearest Micro-Cluster in order to improve concept drift detection and feature tracking. This is illustrated in Figure 5.3 which shows a flowchart of *LPF*.

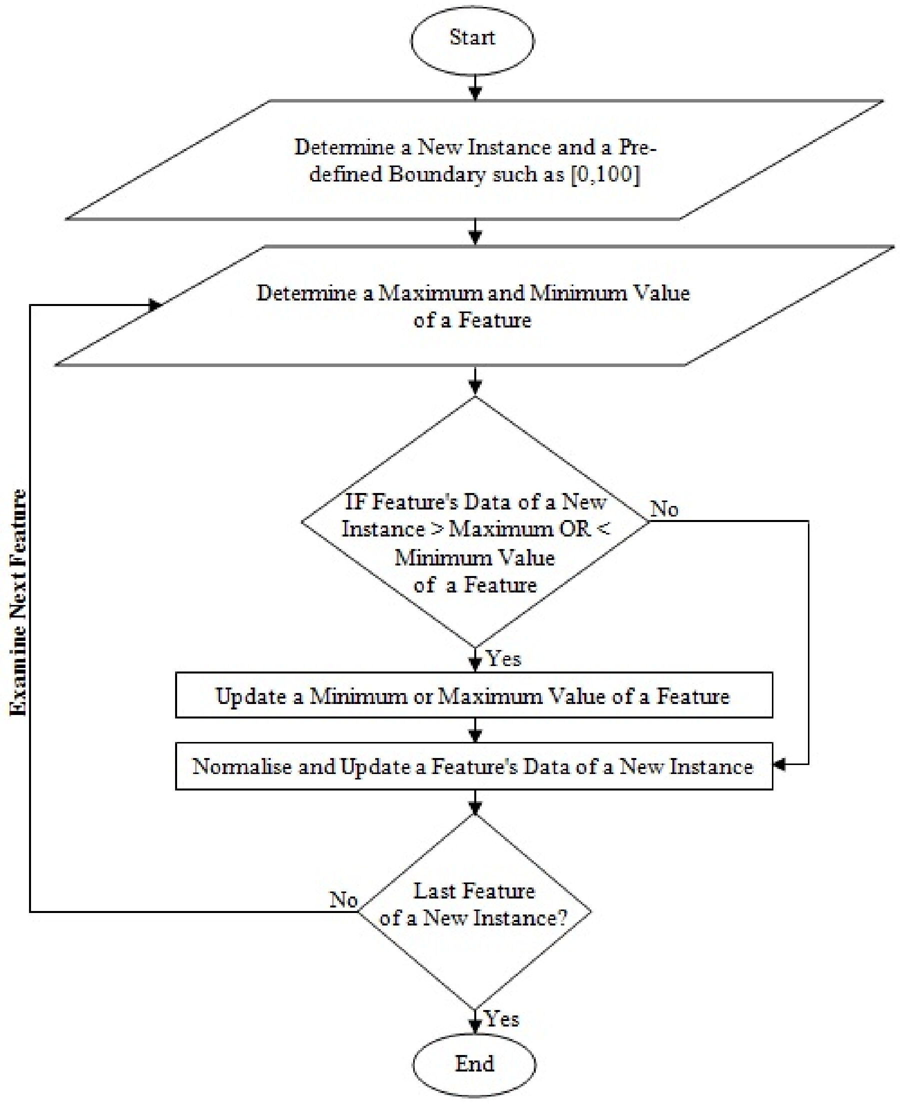
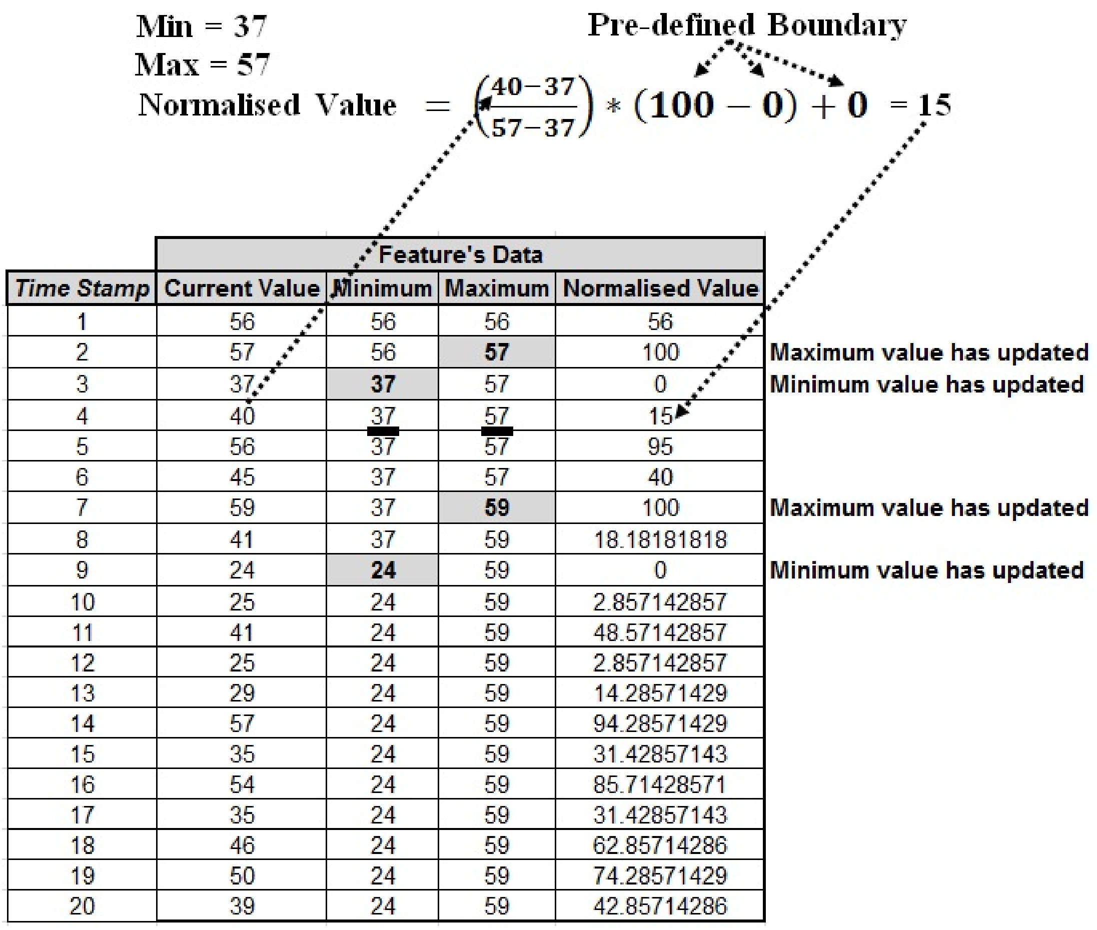

Start
Determine a New Instance and a Pre-defined Boundary such as [0,100]
Determine a Maximum and Minimum Value of a Feature
IF Feature's Data of a New Instance > Maximum OR < Minimum Value of a Feature
No
Yes
Update a Minimum or Maximum Value of a Feature
Normalise and Update a Feature's Data of a New Instance
Last Feature of a New Instance?
No
Yes
End
Examine Next Feature

Figure 5.1: Flowchart of Min-Max *Normalisation*.



Min = 37
Max = 57

$$\text{Normalised Value} = \left(\frac{40-37}{57-37}\right) * (100 - 0) + 0 = 15$$

	Feature's Data				
Time Stamp	Current Value	Minimum	Maximum	Normalised Value	
1	56	56	56	56	
2	57	56	**57**	100	**Maximum value has updated**
3	37	**37**	57	0	**Minimum value has updated**
4	40	37	57	15	
5	56	37	57	95	
6	45	37	57	40	
7	59	37	**59**	100	**Maximum value has updated**
8	41	37	59	18.18181818	
9	24	**24**	59	0	**Minimum value has updated**
10	25	24	59	2.857142857	
11	41	24	59	48.57142857	
12	25	24	59	2.857142857	
13	29	24	59	14.28571429	
14	57	24	59	94.28571429	
15	35	24	59	31.42857143	
16	54	24	59	85.71428571	
17	35	24	59	31.42857143	
18	46	24	59	62.85714286	
19	50	24	59	74.28571429	
20	39	24	59	42.85714286	

Figure 5.2: An example of Min-Max *Normalisation* in real-time.

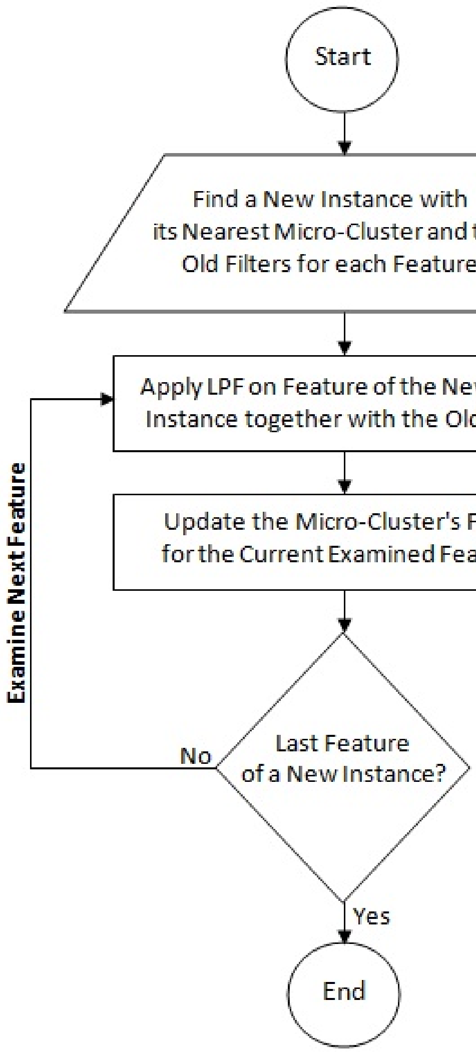
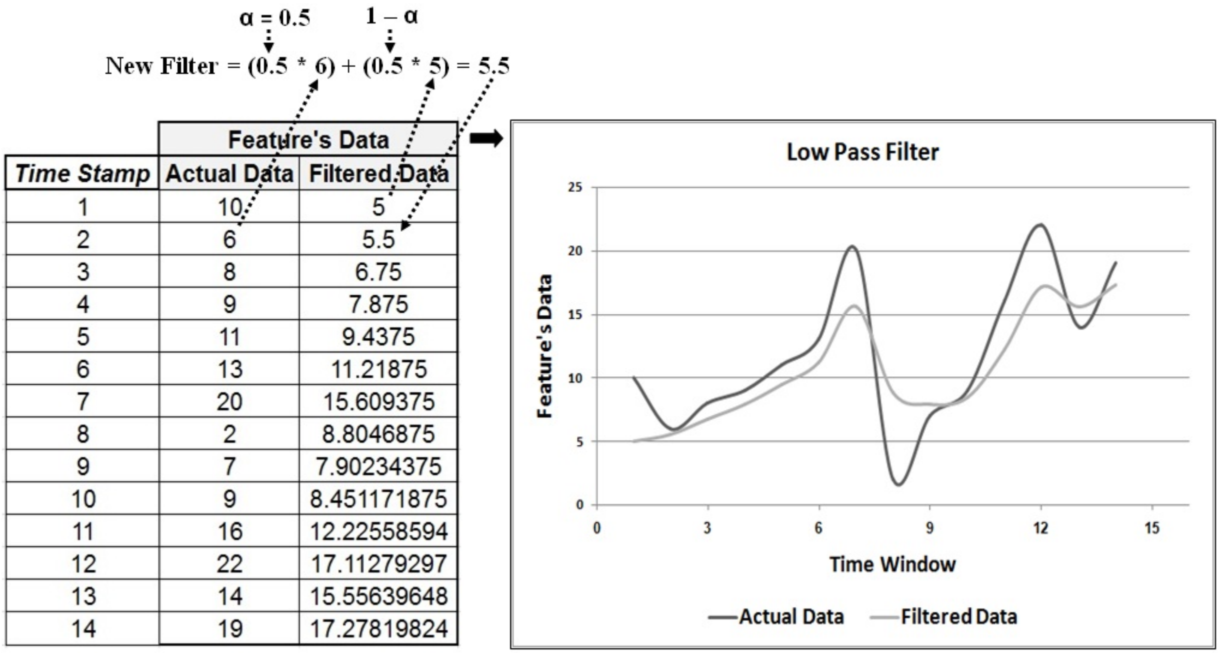

Start
Find a New Instance with its Nearest Micro-Cluster and the Old Filters for each Feature
Apply LPF on Feature of the New Data Instance together with the Old Filter
Update the Micro-Cluster's Filter for the Current Examined Feature
Last Feature of a New Instance?
No
Yes
Examine Next Feature
End

Figure 5.3: Flowchart of *LPF*.

LPF is given by the Equation ($new\ filter[v] = \alpha * new\ value[v] + (1-\alpha) * old\ filter[v]$). v is a feature value of the new data instance. $new\ filter[v]$ and $old\ filter[v]$ can be re-initialised over time stamps as the buffering of data is not required. The α threshold is set to 0.5 (50%) by default, as examined in Appendix A. However, the user may change this to a different threshold. Figure 5.4 shows an example of *LPF*. The *centroid* of each filtered feature of a Micro-Cluster are calculated. Micro-Clusters with normalised and filtered features are passed to the concept drift detection method which was described in Chapter 4.



	Feature's Data	
Time Stamp	Actual Data	Filtered Data
1	10	5
2	6	5.5
3	8	6.75
4	9	7.875
5	11	9.4375
6	13	11.21875
7	20	15.609375
8	2	8.8046875
9	7	7.90234375
10	9	8.451171875
11	16	12.22558594
12	22	17.11279297
13	14	15.55639648
14	19	17.27819824

Figure 5.4: An example of *LPF*.

5.3 Real-Time Feature Tracking

This section describes real-time feature tracking measuring a feature's *Velocity* and spread using both *Variance* and Inter Quartile Range (*IQR*) which are derived from the captured statistics of the Micro-Cluster. Where, the structure of Micro-Clusters is:
$< CF2^x, CF1^x, CF1^t, n, CL, \varepsilon, \Theta, \alpha, \Omega >$. Details about the Micro-Cluster structure components are given in Chapter 3 Table 3.1.

5.3.1 Velocity of Features

Velocity can be tracked through an extension of the MC-NN Micro-Cluster structure by:
$< CF1^{hx}, n_h >$. Where, these components are equivalent to $CF1^x$ and n which were described in Chapter 3. However, the h denotes that these components are *historical* summaries (taken from the previous time stamp); previous *centroid* of a feature x of a Micro-Cluster. The *Velocity* of x can then be calculated using Equation 5.1 after each absorbing of a new instance to its nearest Micro-Cluster (see Chapter 3 Section 3.1), i.e., over time stamps. Figure 5.5 shows an example of feature *Velocity* with a Micro-Cluster consisting of two features.

$$Velocity[x] = \mid \frac{CF1^x}{n} - \frac{CF1^{h,x}}{n_h} \mid \tag{5.1}$$

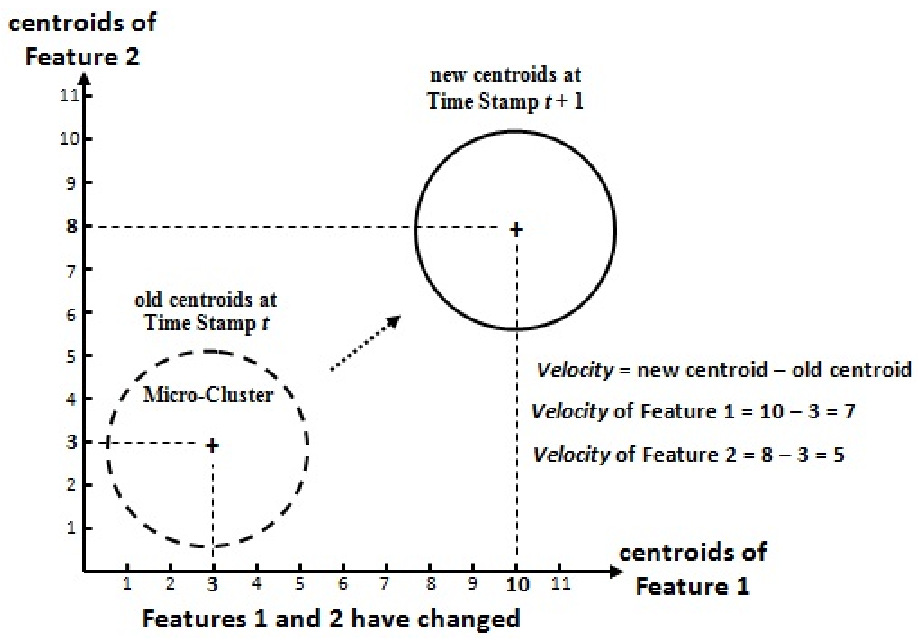

centroids of Feature 2
new centroids at Time Stamp $t+1$
old centroids at Time Stamp t
Micro-Cluster
Velocity = new centroid – old centroid
Velocity of Feature 1 = 10 – 3 = 7
Velocity of Feature 2 = 8 – 3 = 5
centroids of Feature 1
Features 1 and 2 have changed
1 2 3 4 5 6 7 8 9 10 11

Figure 5.5: An example of feature *Velocity*.

5.3.2 Measuring of the Data Stream Spread using IQR

In the original MC-NN, the spread of the values absorbed in a Micro-Cluster is measured using *Variance*. The spread quantifies how varied the set of a feature's values are. However, the performance of *Variance* can be affected negatively by outliers as they may significantly change the μ. *IQR* is considered a more robust method with respect to outliers (Leys et al., 2013; Sunitha et al., 2014) compared with *Variance*. The first version of the here presented feature tracker used *Variance* as a measure of a feature's spread. However, as expected it was observed that this approach did not work very well due to its sensitivity to outliers. Thus *IQR* was explored as an alternative to *Variance* in order to obtain a more reliable measure of the Micro-Cluster spread. The evaluation results presented in Section 5.4 compare the performance of the proposed feature tracker as obtained using either *Variance* or *IQR*. This chapter describes the new extension of the MC-NN Micro-Cluster structure with *IQR* for the purpose of splitting a Micro-Cluster which is explained in the next section.

5.3.3 Splitting of a Micro-Cluster using IQR

One purpose of using *IQR* in this research is to split a Micro-Cluster once its error counts ε reaches Θ. The use of ε and Θ was described in Chapter 3. Again original MC-NN uses *Variance* and this research prefers *IQR* as it is more robust to outliers. The assumption here is that the larger the *IQR* of a feature, the greater the range of values that have been seen for that feature and thus it may contribute to mis-absorption of new data instances.

The new Mirco-Clusters resulting from the *Split* are generated with *Q1* and *Q3* quartiles. The *centroids* of the new Micro-Clusters are replaced with either *Q1* or *Q3* in all dimensions (features). In specific one Micro-Cluster only uses *Q1* values and the other only uses *Q3* values. This is shown for one dimension only in Figure 5.6. The original Micro-Cluster is deleted after the *Split* is performed.

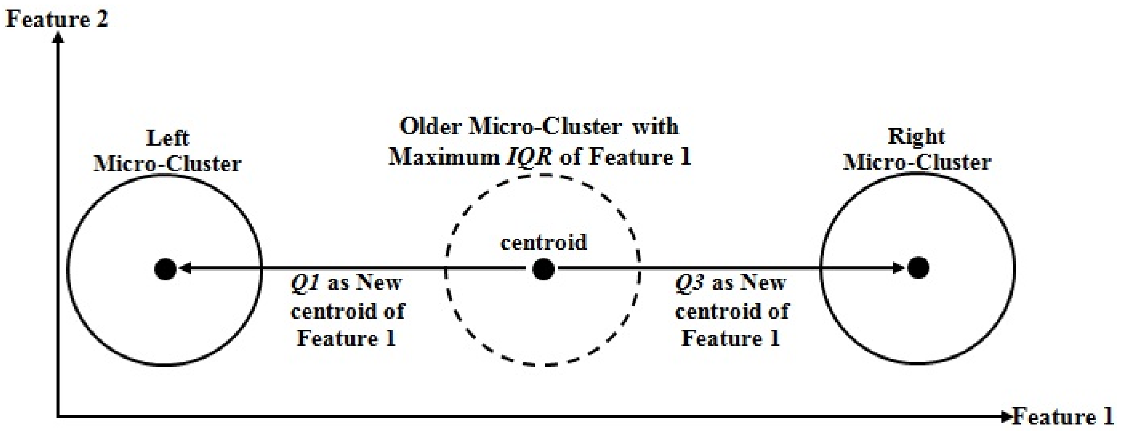

Feature 2
Left Micro-Cluster
Older Micro-Cluster with Maximum IQR of Feature 1
Right Micro-Cluster
centroid
Q1 as New centroid of Feature 1
Q3 as New centroid of Feature 1
Feature 1

Figure 5.6: Splitting of a Micro-Cluster with *IQR*.

Again the results presented in Chapter 4 compare a version of the proposed approach only using *Variance* and the other using *IQR*. The next section provides a working example on Micro-Cluster splitting for both using *Variance* and *IQR*.

5.3.4 Comparison of Splitting using IQR or Variance

This section compares the splitting of a Micro-Cluster containing an outlier using *IQR* and *Variance*. This is to demonstrate that *IQR* is a more robust metric to split Micro-Clusters if there are outliers. For simplicity of this illustration the *Split* along only one feature (dimension) is observed, as shown in the example and Figure 5.7 below with an outlier.

Consider a feature's values $[x] = 1, 1, 1.2, 2.5, 2.8, 3, 4.6, 5, 3000$, and consider the value 3000 to be an outlier. Then $centroid[x] = \mu = (1+1+1.2+2.5+2.8+3+4.6+5+3000)/9 = 335.6778$.

- ***Splitting of centroid with respect to x with IQR*:**

Median = 2.8,

$Q1[x] = (1 + 1.2)/2 = 1.1$,

$Q3[x] = (4.6 + 5)/2 = 4.8$,

$IQR[x] = Q3 - Q1 = 3.7$,

$centroid[x]$ of a Left Micro-Cluster = $Q1[x] = 1.1$,

$centroid[x]$ of a Right Micro-Cluster = $Q3[x] = 4.8$

- ***Splitting of centroid with respect to** x **with** Variance***:

$\left(\frac{CF2^x}{n}\right) = (1^2+1^2+1.2^2+2.5^2+2.8^2+3^2+4.6^2+5^2+3000^2)/9 = 1000008.077$,

$\left(\frac{CF1^x}{n}\right)^2 = ((1+1+1.2+2.5+2.8+3+4.6+5+3000)/9)^2 = 112679.5705$,

$Variance[x] = \sqrt{\left(\frac{CF2^x}{n}\right) - \left(\frac{CF1^x}{n}\right)^2} = 941.9815$,

$centroid[x]$ of a Negative Micro-Cluster = old $centroid[x]$ - $Variance[x]$ = $335.6778 - 941.9815 = -606.3037$,

$centroid[x]$ of a Positive Micro-Cluster = old $centroid[x]$ + $Variance[x]$ = $335.6778 + 941.9815 = 1277.6593$.

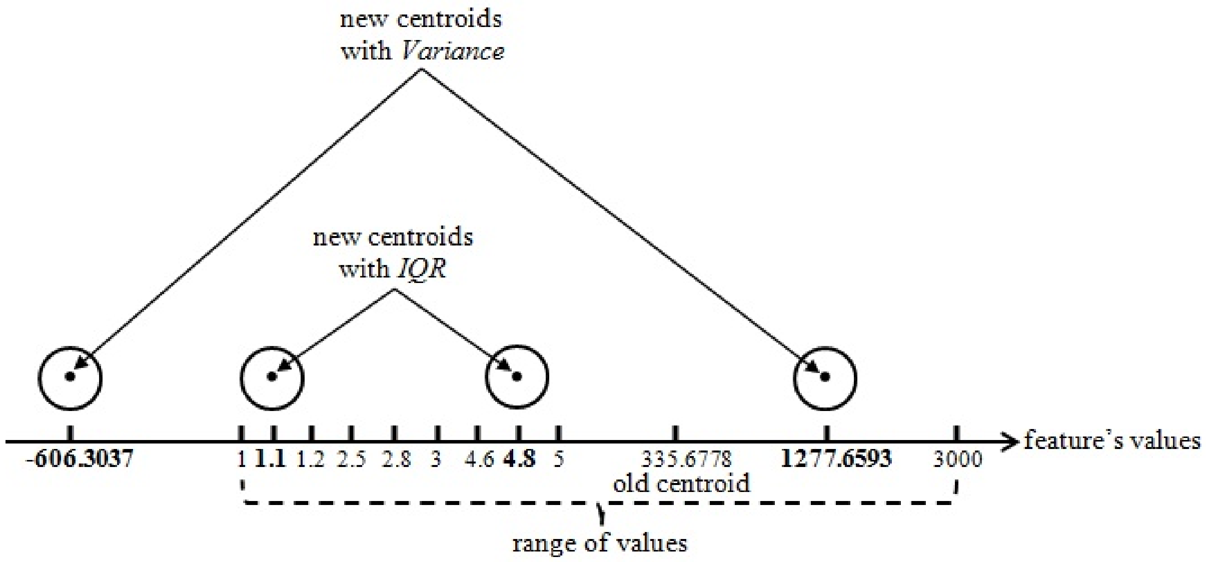


Figure 5.7: An example of splitting a Micro-Cluster with *Variance* and *IQR*.

In the example given above which illustrated in Figure 5.7, it can be seen that the original Micro-Cluster *centroid* is strongly influenced by the outlier. However, it can also be seen that the new *centroids* after splitting using *IQR* are within the value range of the features values (excluding the outlier), whereas the new *centroids* after splitting using *Variance*, as it is performed in original MC-NN (see Chapter 3), are shifted towards the outlier outside the original

Micro-Cluster's value range. Thus, in this research *IQR* is used to perform Micro-Cluster splits to mediate the influence of outliers.

In order to use the *IQR*, the feature's data values need to be ordered or sorted in ascending order. To achieve this in real-time (i.e., a computationally efficient manner), *First-In-First-Out* ($FIFO$) queue principle combined with *SkipList* is used which is explained in the next section.

5.3.5 First-In-First-Out (FIFO) Queue with SkipList

In order to identify the quartiles (i.e., $Q1$ and $Q3$), some of the feature's values have to be saved and sorted in ascending order in real-time. Firstly, in order to save the feature data values a *First-In-First-Out* ($FIFO$) queue is applied. Where the oldest entry of the queue is handled first. $FIFO$'s size can be set by a user threshold which is less or equal to the size of the statistical windows (time windows). $FIFO$ keeps the most recent data (i.e., feature's data) and needs to be set to a lowest possible size (i.e., less or equal to a window's size) which yields good results. In an initial feasibility study, it was found that a queue size of 1000 works well in most cases regarding computational efficiency and accuracy as examined in Appendix B. Thus a queue of size 1000 has been used in all experiments presented in this research. It was also observed in most cases that even if the statistical window size is much larger than 1000, that it is unlikely that a single Micro-Cluster absorbs more than 1000 instances. Thus the potential loss of information due to a queue size limit is also unlikely.

Secondly, for sorting the $FIFO$ queue of a feature x, a *SkipList* is used, which is a data structure that enables fast search and insert $O(\log n)$ rather than $O(n)$ operations by updating a linked hierarchy of sub-sequences within an ordered sequence of elements (Shavit and Lotan, 2000; De Gregorio and Di Stefano, 2017). *SkipList* is superior to alternative sorting algorithms regarding computational efficiency as indicated in (Hu et al., 2003). Each node in a *SkipList* consists of four directions (i.e., up, down, left, and right), as shown in Figure 5.8. Once splitting of a Micro-Cluster with larger or maximum *IQR* is performed, the $FIFO$ queue of each feature x of a Micro-Cluster is then sent to the $SkipList$.

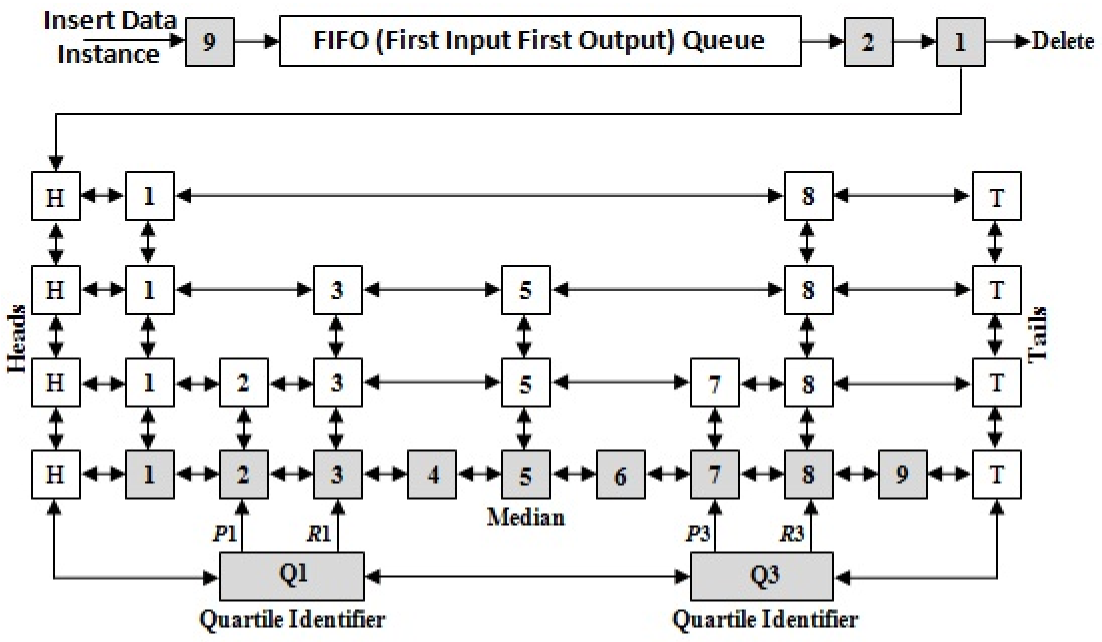


Figure 5.8: Real-time sorting of the feature data values using a *SkipList*.

In order to locate and update the position of $Q1$ and $Q3$, *Quartile Identifiers* are created, which are data structure connected with the head and tail of a *SkipList* as illustrated in Figure 5.8. A *Quartile Identifier* consists of two nodes P and R, P is a pointer to either the value next to the left of the quartile $Q1$ or $Q3$, and R is a pointer to the next value right of P. The index of P and R is either 1 or 3, which denotes if the pointers refer to Quartile 1 or 3 respectively. The *Quartile Identifier* skips either to the left or right position after each insertion of a new *SkipList* node , as shown in Algorithms 2, 3, and 4. *Quartile Identifier* of $Q1$ is located in the middle of the left half of *median* of the *SkipList* and connected with the head of the *SkipList* indicated with letter H in Figure 5.8. While the *Quartile Identifier* of $Q3$ is located in the middle of the right half and connected with the tail of the *SkipList* indicated with letter T in Figure 5.8.

Given P and R of a *Quartile Identifier* and if the number of feature values left/right of the *median* is either even or odd, then quartiles $Q1$ and $Q3$ can be calculated using the following equations. Where, y refers to either $Q1$ or $Q3$. If the number of feature values left/right of the *median* is even then Equation 5.2 is used and if it is odd then Equation 5.3 is used.

$$Q_y[x] = (P_y + R_y)/2 \tag{5.2}$$

$$Q_y[x] = P_y \tag{5.3}$$

When adjusting the *Quartile Identifiers* in real-time there are three possible cases to consider depending on whether the total number of values of the feature is odd or even and also if the number of values to the left and right of the *median* is odd or even. For each of the three cases, three further scenarios exist depending. Where the value is inserted in the *SkipList*. These cases and scenarios are described in the remainder of this chapter including examples.

Case 1: *After inserting the new value, the total number of feature values in the SkipList is even, and the number of values left or right of the median is odd.*

In this specific case Algorithm 2 is applied.

Algorithm 2 Adjusting *Quartile Identifiers* for lower and upper quartiles ($Q1$ and $Q3$) in terms of **Case 1**.

Input: new feature value f, nodes of a *SkipList* after adding a new feature value f, previous *Quartile Identifiers* $P1$ and $P3$.
Output: updated *Quartile Identifiers* $P1$ and $P3$

```
1:  if $f \leq value\ of\ P1$ then
2:      $P1$ and $P3$ remain at the same position
3:  else
4:      if $f > value\ of\ P1\ and\ f \leq value\ of\ P3$ then
5:          Skip $P1$ to the next value right of $P1$
6:      else
7:          if $f > value\ of\ P3$ then
8:              Skip $P1$ to the next value right of $P1$
9:              Skip $P3$ to the next value right of $P3$
10:         end if
11:     end if
12: end if
```

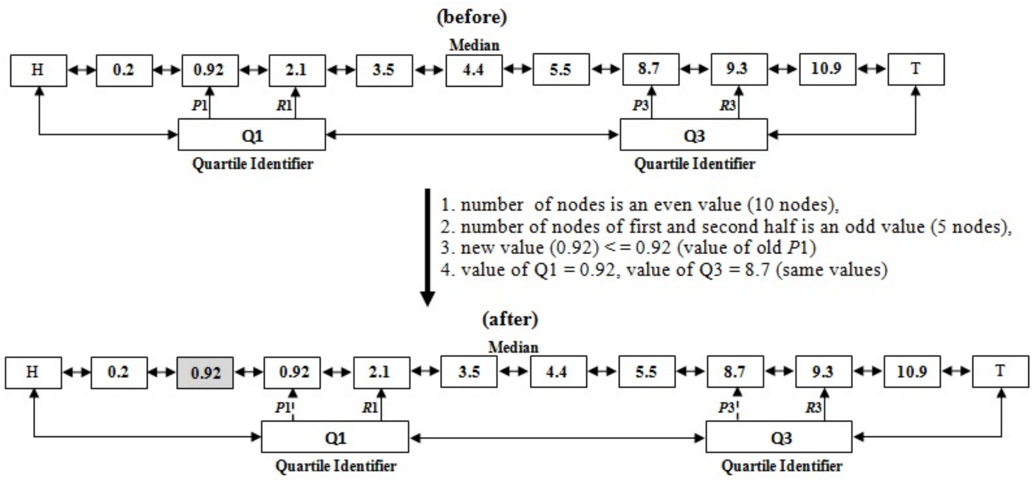

(before)
Median
H
0.2
0.92
2.1
3.5
4.4
5.5
8.7
9.3
10.9
T
P1
R1
P3
R3
Q1
Quartile Identifier
Q3
Quartile Identifier
1. number of nodes is an even value (10 nodes),
2. number of nodes of first and second half is an odd value (5 nodes),
3. new value (0.92) <= 0.92 (value of old P1)
4. value of Q1 = 0.92, value of Q3 = 8.7 (same values)
(after)
Median
H
0.2
0.92
0.92
2.1
3.5
4.4
5.5
8.7
9.3
10.9
T
P1'
R1
P3'
R3
Q1
Quartile Identifier
Q3
Quartile Identifier

Figure 5.9: An example of the update for the *Quartile Identifiers* for Case 1 with a value inserted left of *Q1*.

In Figure 5.9, the number of nodes of the *SkipList* for the feature is an even value (10 nodes) after adding the new feature value 0.92, and the number of nodes of the first and second half of the *SkipList* is an odd value (5 nodes). The new value $f \leq value\ of\ P1$, thus lines 1 and 2 in Algorithm 2 are executed and $Q1$ and $Q3$ are computed, in this scenario $P1$ and $P3$ remain unchanged. The Quartiles are computed using Equation 5.3.

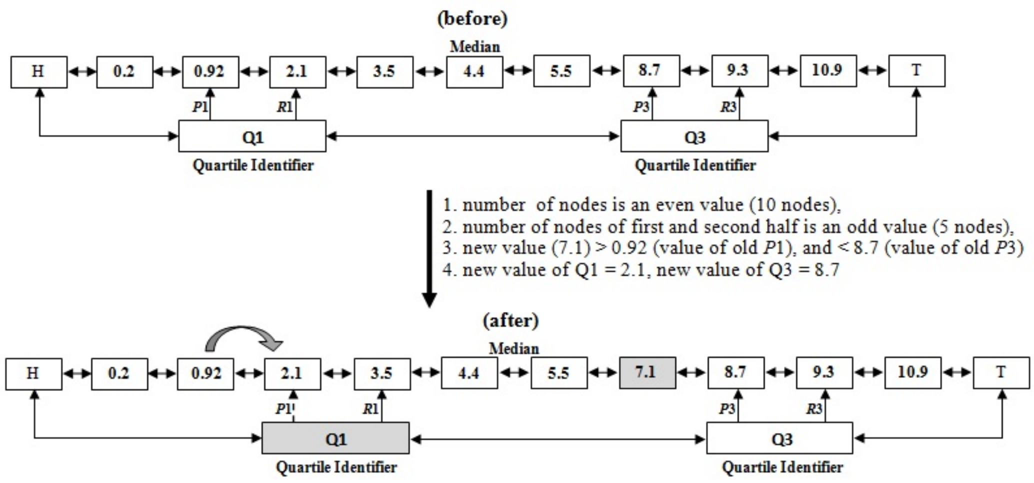

(before)
Median
H
0.2
0.92
2.1
3.5
4.4
5.5
8.7
9.3
10.9
T
P1
R1
P3
R3
Q1
Quartile Identifier
Q3
Quartile Identifier
1. number of nodes is an even value (10 nodes),
2. number of nodes of first and second half is an odd value (5 nodes),
3. new value (7.1) > 0.92 (value of old P1), and < 8.7 (value of old P3)
4. new value of Q1 = 2.1, new value of Q3 = 8.7
(after)
Median
H
0.2
0.92
2.1
3.5
4.4
5.5
7.1
8.7
9.3
10.9
T
P1
R1
P3
R3
Q1
Quartile Identifier
Q3
Quartile Identifier

Figure 5.10: An example of the update for the *Quartile Identifiers* for Case 1 with a value inserted between of *Q1* and *Q3*.

In Figure 5.10, the number of nodes of the *SkipList* for the feature is an even value (10 nodes) after adding the new feature value 7.1, and the number of nodes of the first and second half of the *SkipList* is an odd value (5 nodes). The new value $f > value\ of\ P1\ and\ f \leq value\ of\ P3$, thus lines 4 and 5 in Algorithm 2 are executed and $Q1$ and $Q3$ are computed. In this case $P1$ is skipped to the next value right of $P1$ and $P3$ remains unchanged. The Quartiles are computed using Equation 5.3.

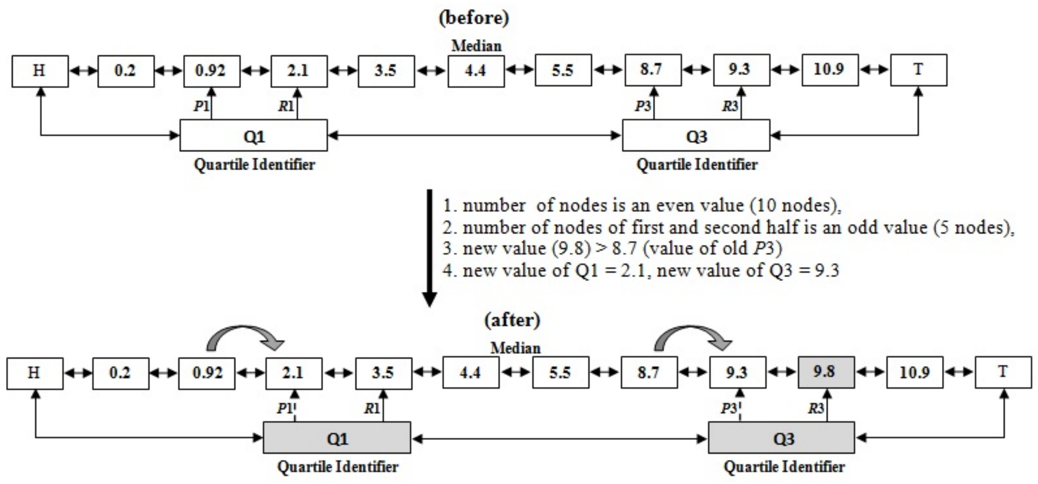

(before)
Median
H
0.2
0.92
2.1
3.5
4.4
5.5
8.7
9.3
10.9
T
P1
R1
P3
R3
Q1
Q3
Quartile Identifier
Quartile Identifier
1. number of nodes is an even value (10 nodes),
2. number of nodes of first and second half is an odd value (5 nodes),
3. new value (9.8) > 8.7 (value of old P3)
4. new value of Q1 = 2.1, new value of Q3 = 9.3
(after)
Median
H
0.2
0.92
2.1
3.5
4.4
5.5
8.7
9.3
9.8
10.9
T
P1
R1
P3
R3
Q1
Q3
Quartile Identifier
Quartile Identifier

Figure 5.11: An example of the update for the *Quartile Identifiers* for Case 1 with a value inserted right of *Q3*.

In Figure 5.11, the number of nodes of the *SkipList* for the feature is an even value (10 nodes) after adding the new feature value 9.8, and the number of nodes of the first and second half of the *SkipList* is an odd value (5 nodes). The new value $f > value\ of\ P3$, thus lines 7 to 10 in Algorithm 2 are executed and $Q1$ and $Q3$ are computed. In this case $P1$ and $P3$ are skipped to the next value right of $P1$ and $P3$ respectively. The Quartiles are computed using Equation 5.3.

Case 2: *After inserting the new value, the total number of feature values in the SkipList is an odd value.*

In this specific case Algorithm 3 is applied.

Algorithm 3 Adjusting *Quartile Identifiers* for lower and upper quartiles ($Q1$ and $Q3$) in terms of **Case 2**.

Input: new feature value f, nodes of a *SkipList* after adding a new feature value f, previous *Quartile Identifiers* $P1$ and $P3$.
Output: updated *Quartile Identifiers* $P1$ and $P3$

```
1:  if $f \leq value\ of\ P1$ then
2:      Skip $P1$ to the next value left of $P1$
3:  else
4:      if $f > value\ of\ P1\ and\ f \leq value\ of\ P3$ then
5:          $P1$ and $P3$ remain at the same position
6:      else
7:          if $f > value\ of\ P3$ then
8:              Skip $P3$ to the next value right of $P3$
9:          end if
10:     end if
11: end if
```

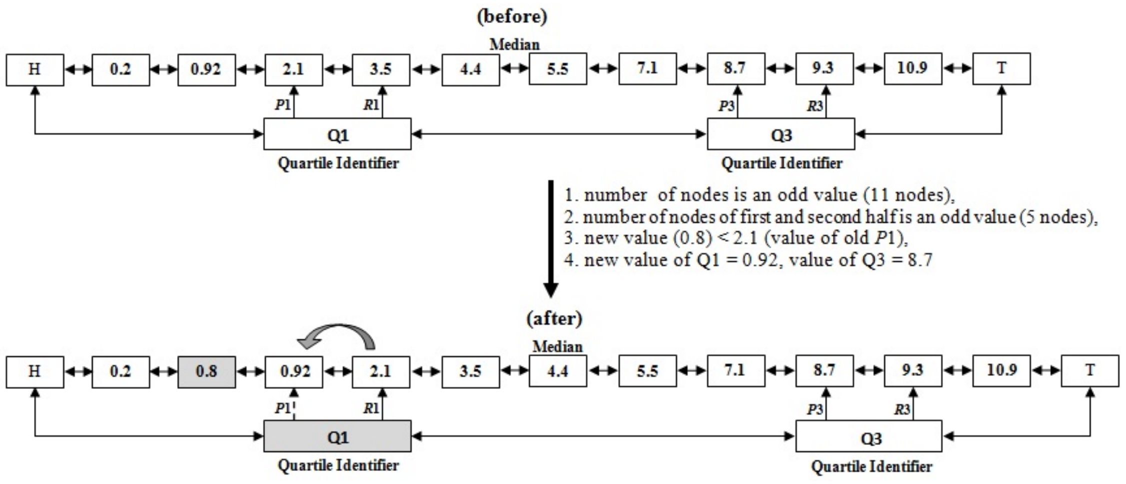

(before)
Median
H 0.2 0.92 2.1 3.5 4.4 5.5 7.1 8.7 9.3 10.9 T
P1 R1 P3 R3
Q1
Quartile Identifier
Q3
Quartile Identifier
1. number of nodes is an odd value (11 nodes),
2. number of nodes of first and second half is an odd value (5 nodes),
3. new value (0.8) < 2.1 (value of old P1),
4. new value of Q1 = 0.92, value of Q3 = 8.7
(after)
Median
H 0.2 0.8 0.92 2.1 3.5 4.4 5.5 7.1 8.7 9.3 10.9 T
P1 R1 P3 R3
Q1
Quartile Identifier
Q3
Quartile Identifier

Figure 5.12: An example of the update for the *Quartile Identifiers* for Case 2 with a value inserted left of *Q1*.

In Figure 5.12, the number of nodes of the *SkipList* for the feature is an odd value (11 nodes) after adding the new feature value 0.8. The new value $f \leq value\,of\,P1$, thus lines 1 and 2 in Algorithm 3 are executed and $Q1$ and $Q3$ are computed. In this case $P1$ is skipped to the next value left of $P1$ and $P3$ remains unchanged. The Quartiles are computed using Equation 5.3.

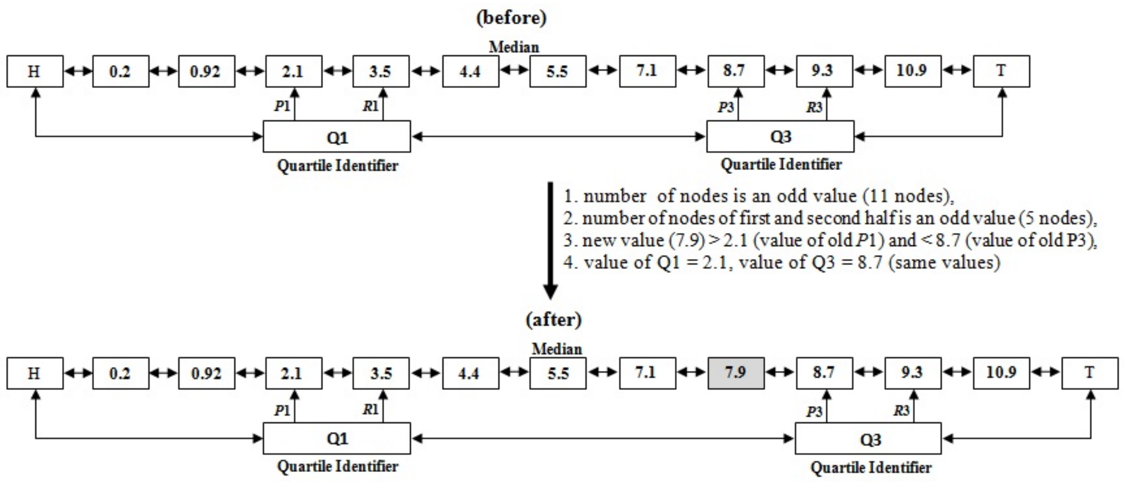

(before)
Median
H 0.2 0.92 2.1 3.5 4.4 5.5 7.1 8.7 9.3 10.9 T
P1 R1 P3 R3
Q1
Quartile Identifier
Q3
Quartile Identifier
1. number of nodes is an odd value (11 nodes),
2. number of nodes of first and second half is an odd value (5 nodes),
3. new value (7.9) > 2.1 (value of old P1) and < 8.7 (value of old P3),
4. value of Q1 = 2.1, value of Q3 = 8.7 (same values)
(after)
Median
H 0.2 0.92 2.1 3.5 4.4 5.5 7.1 7.9 8.7 9.3 10.9 T
P1 R1 P3 R3
Q1
Quartile Identifier
Q3
Quartile Identifier

Figure 5.13: An example of the update for the *Quartile Identifiers* for Case 2 with a value inserted between *Q1* and *Q3*.

In Figure 5.13, the number of nodes of the *SkipList* for the feature is an odd value (11 nodes) after adding the new feature value of 7.9. The new value $f > value\,of\,P1\ and\ f \leq value\,of\,P3$, thus lines 4 and 5 in Algorithm 3 are executed and $Q1$ and $Q3$ are computed. In this case $P1$ and $P3$ remains unchanged. The Quartiles are computed using Equation 5.3.

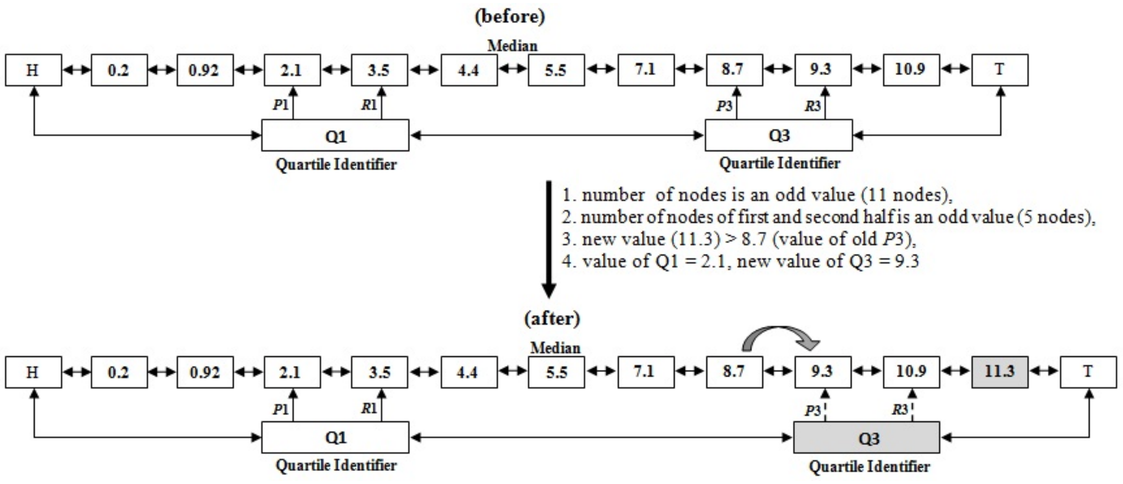

(before)
Median
H
0.2
0.92
2.1
3.5
4.4
5.5
7.1
8.7
9.3
10.9
T
P1
R1
P3
R3
Q1
Quartile Identifier
Q3
Quartile Identifier
1. number of nodes is an odd value (11 nodes),
2. number of nodes of first and second half is an odd value (5 nodes),
3. new value (11.3) > 8.7 (value of old P3),
4. value of Q1 = 2.1, new value of Q3 = 9.3
(after)
Median
11.3
T

Figure 5.14: An example of the update for the *Quartile Identifiers* for Case 2 with a value inserted right of *Q3*.

In Figure 5.14, the number of nodes of the *SkipList* for the feature is an odd value (11 nodes) after adding the new feature value 11.3. The new value $f > value\ of\ P3$, thus lines 7 to 9 in Algorithm 3 are executed and $Q1$ and $Q3$ are computed. In this case $P3$ is skipped to the next value right of $P3$ and $P1$ remains unchanged. The Quartiles are computed using equation 5.3.

Case 3: *After inserting the new value, the total number of feature values in the SkipList is even, and the number of values left or right of the median is even.*

In this specific case Algorithm 4 is applied.

Algorithm 4 Adjusting *Quartile identifiers* for lower and upper quartiles ($Q1$ and $Q3$) in terms of **Case 3**.

Input: new feature value f, nodes of a *SkipList* after adding a new feature value f, previous *Quartile Identifiers* $P1$ and $P3$.

Output: updated *Quartile Identifiers* $P1$ and $P3$

```
1:  if $f \leq value\ of\ P1$ then
2:      Skip $P1$ to the next value left of $P1$
3:      Skip $P3$ to the next value left of $P3$
4:  else
5:      if $f > value\ of\ P1$ and $f \leq value\ of\ P3$ then
6:          Skip $P3$ to the next value left of $P3$
7:      else
8:          if $f > value\ of\ P3$ then
9:              $P1$ and $P3$ remain at the same position
10:         end if
11:     end if
12: end if
```

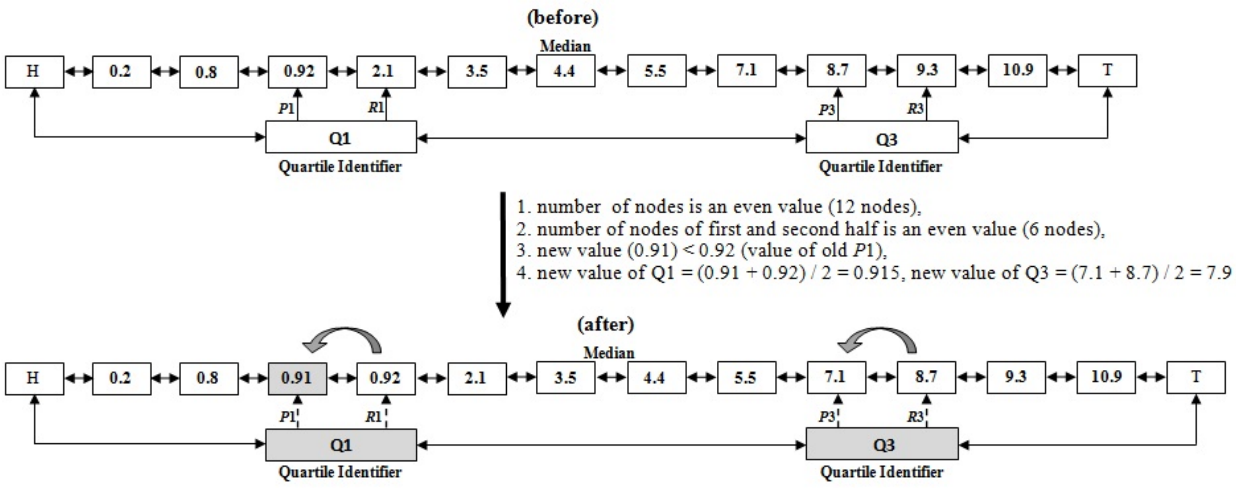

(before)
Median
H
0.2
0.8
0.92
2.1
3.5
4.4
5.5
7.1
8.7
9.3
10.9
T
P1
R1
P3
R3
Q1
Quartile Identifier
Q3
Quartile Identifier
1. number of nodes is an even value (12 nodes),
2. number of nodes of first and second half is an even value (6 nodes),
3. new value (0.91) < 0.92 (value of old P1),
4. new value of Q1 = (0.91 + 0.92) / 2 = 0.915, new value of Q3 = (7.1 + 8.7) / 2 = 7.9
(after)
0.91

Figure 5.15: An example of the update for the *Quartile Identifiers* for Case 3 with a value inserted left of *Q1*.

In Figure 5.15, the number of nodes of the *SkipList* for the feature is an even value (12 nodes) after adding the new feature value of 0.91. The new value $f \leq value\,of\,P1$, thus lines 1 to 3 in Algorithm 4 are executed and $Q1$ and $Q3$ are computed. In this case $P1$ and $P3$ are skipped to the next value left of $P1$ and $P3$ respectively. The Quartiles are computed using Equation 5.2.

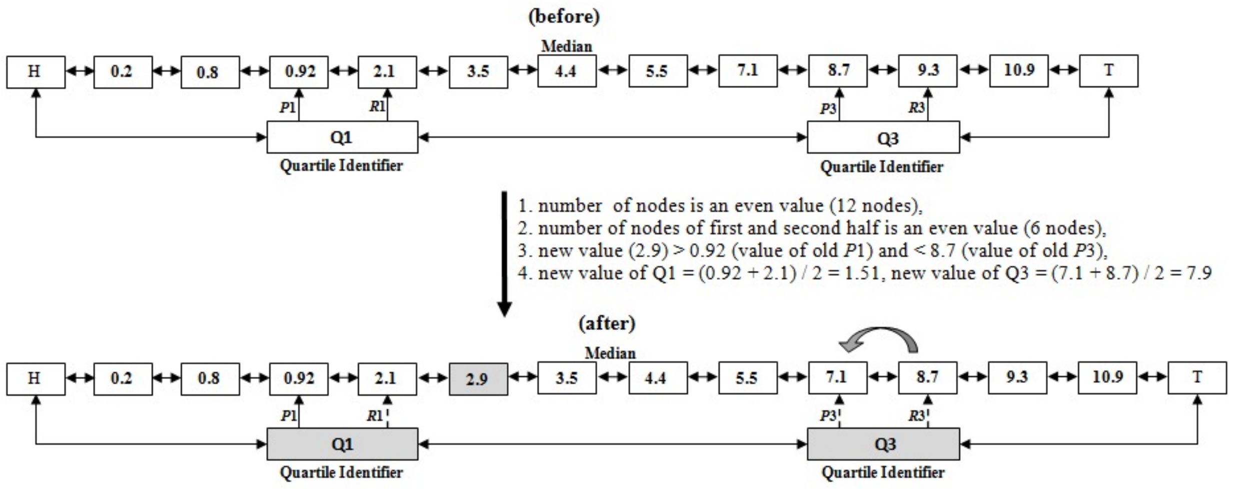

(before)
Median
H
0.2
0.8
0.92
2.1
3.5
4.4
5.5
7.1
8.7
9.3
10.9
T
P1
R1
P3
R3
Q1
Quartile Identifier
Q3
Quartile Identifier
1. number of nodes is an even value (12 nodes),
2. number of nodes of first and second half is an even value (6 nodes),
3. new value (2.9) > 0.92 (value of old P1) and < 8.7 (value of old P3),
4. new value of Q1 = (0.92 + 2.1) / 2 = 1.51, new value of Q3 = (7.1 + 8.7) / 2 = 7.9
(after)
2.9

Figure 5.16: An example of the update for the *Quartile Identifiers* for Case 3 with a value inserted between *Q1* and *Q3*.

In Figure 5.16, the number of nodes of the *SkipList* for the feature is an even value (12 nodes) after adding the new feature value of 2.9. The new value $f > value\,of\,P1\,and\,f \leq value\,of\,P3$, thus lines 5 and 6 in Algorithm 4 are executed and $Q1$ and $Q3$ are computed. In this case $P3$ is skipped going to the next value left of $P3$ and $P1$ remains unchanged. The Quartiles are computed using Equation 5.2.

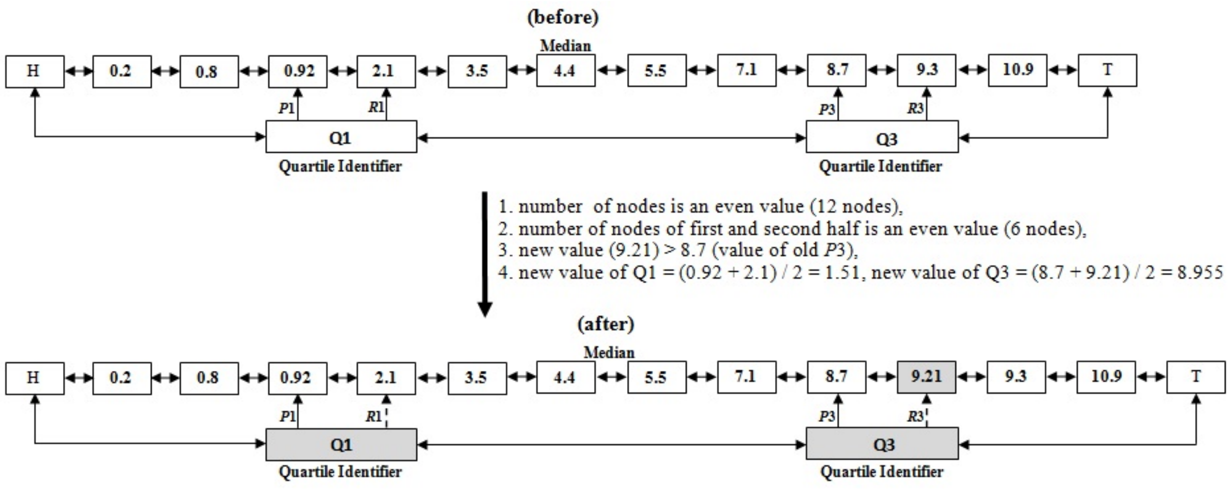

(before)
Median
H
0.2
0.8
0.92
2.1
3.5
4.4
5.5
7.1
8.7
9.3
10.9
T
P1
R1
P3
R3
Q1
Quartile Identifier
Q3
Quartile Identifier
1. number of nodes is an even value (12 nodes),
2. number of nodes of first and second half is an even value (6 nodes),
3. new value (9.21) > 8.7 (value of old P3),
4. new value of Q1 = (0.92 + 2.1) / 2 = 1.51, new value of Q3 = (8.7 + 9.21) / 2 = 8.955
(after)
Median
H
0.2
0.8
0.92
2.1
3.5
4.4
5.5
7.1
8.7
9.21
9.3
10.9
T
P1
R1
P3
R3
Q1
Quartile Identifier
Q3
Quartile Identifier

Figure 5.17: An example of the update for the *Quartile Identifiers* for Case 3 with a value inserted right of *Q3*.

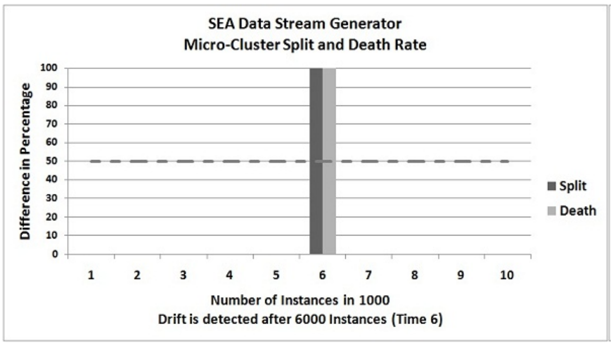
In Figure 5.17, the number of nodes of the *SkipList* for the feature is an even value (12 nodes) after adding the new feature value of 9.21. The new value $f > value\,of\,P3$, thus lines 8 to 10 in Algorithm 4 are executed and $Q1$ and $Q3$ are computed. In this case $P1$ and $P3$ remain unchanged. The Quartiles are computed using Equation 5.2.

5.4 Empirical Evaluation of Real-Time Feature Tracking Method using Variance and IQR

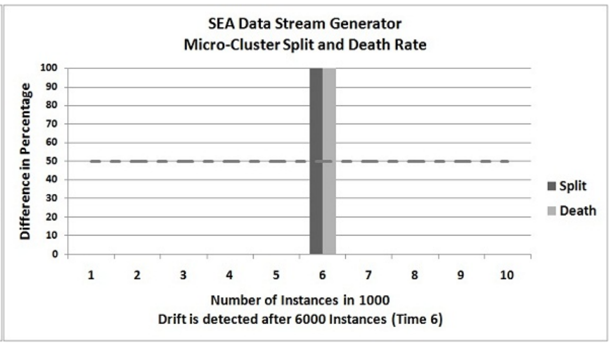
The evaluation presented in the previous chapter was based on original MC-NN using the readily available *Variance* statistics of the Micro-Clusters and *Split* and *Death* rates to detect concept drift. It has been found that the developed concept drift detection method was affected by feature-bias, noise, and outliers. This section applies *Normalisation* and *LPF* in real-time in order to address these effects and compares these results with those from Section 4.3. However, the benefit of applying *Normalisation* in combination with *LPF* in terms of *Velocity* is examined in Appendix G as this is not considered the main aim of this section. Where this section compares the previously discussed method for feature tracking based on *Variance* combined with feature *Velocity* to the more robust method based on *IQR* combined with feature *Velocity*. For the experiments the default parameters stated in Table 3.2 (artificial datasets) of the method were used unless stated otherwise. Regarding the artificial datasets, no *Normalisation* was applied as the data stream generators already produced normalised data.

Part (a) of Figures 5.18 to 5.32 shows the results for using MC-NN with *Variance* and history

(a) Previous MC-NN
(LPF and **Variance**)



(b) New MC-NN
(LPF and **IQR**)



Features are ordered based on *Velocity* rate

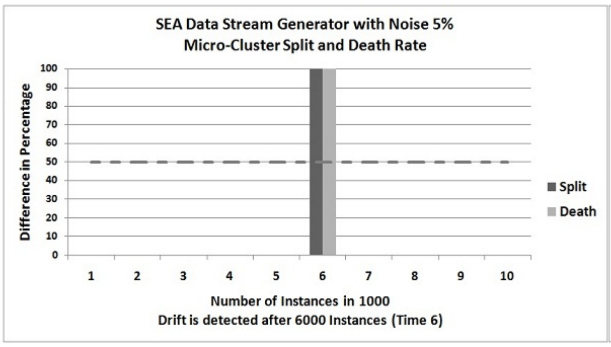
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 2 and 3)
Feature 2	**0.01221**	**2**	True Positive
Feature 3	**0.00997**	**1**	True Positive
Feature 1	0.00202	1	

Features are ordered based on *Velocity* rate

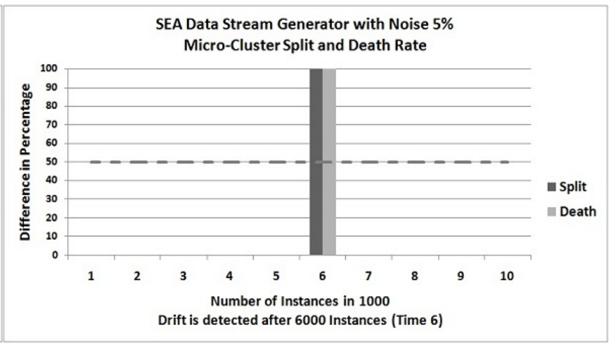
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 2 and 3)
Feature 3	**0.00229**	**3**	True Positive
Feature 2	**0.00158**	**2**	True Positive
Feature 1	0.00118	0	

Figure 5.18: The results of *SEA* data stream generator with a noise level of 0% using Micro-Clusters for tracking features.

(a) Previous MC-NN
(LPF and **Variance**)



(b) New MC-NN
(LPF and **IQR**)



Features are ordered based on *Velocity* rate

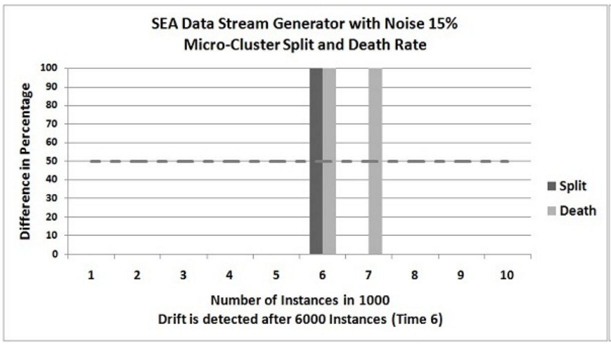
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 2 and 3)
Feature 3	**0.00421**	**1**	True Positive
Feature 2	0.00397	0	False Negative
Feature 1	0.00317	0	

Features are ordered based on *Velocity* rate

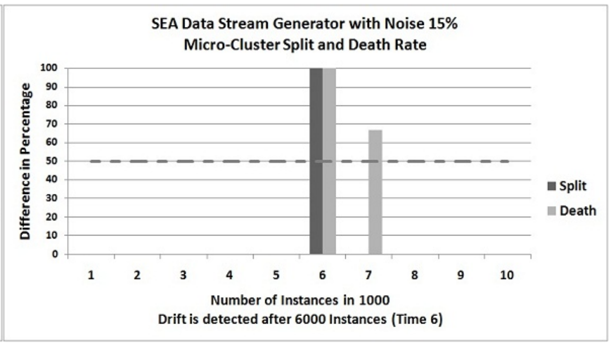
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 2 and 3)
Feature 2	**0.00143**	**1**	True Positive
Feature 3	**0.00136**	**1**	True Positive
Feature 1	0.00033	0	

Figure 5.19: The results of *SEA* data stream generator with a noise level of 5% using Micro-Clusters for tracking features.

(a) Previous MC-NN
(LPF and **Variance**)



(b) New MC-NN
(LPF and **IQR**)



Features are ordered based on *Velocity* rate

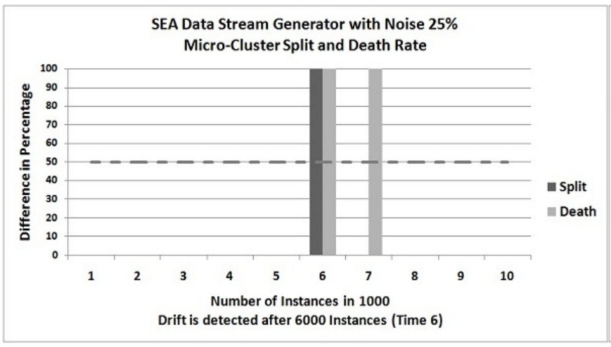
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 2 and 3)
Feature 2	0.00258	0	False Negative
Feature 3	**0.00212**	**1**	True Positive
Feature 1	0.00202	0	

Features are ordered based on *Velocity* rate

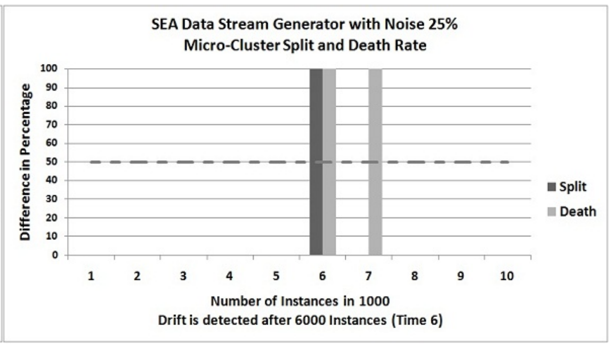
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 2 and 3)
Feature 3	**0.00230**	**2**	True Positive
Feature 1	0.00094	0	
Feature 2	0.00078	0	False Negative

Figure 5.20: The results of *SEA* data stream generator with a noise level of 15% using Micro-Clusters for tracking features.

(a) Previous MC-NN
(LPF and **Variance**)



(b) New MC-NN
(LPF and **IQR**)



Features are ordered based on *Velocity* rate

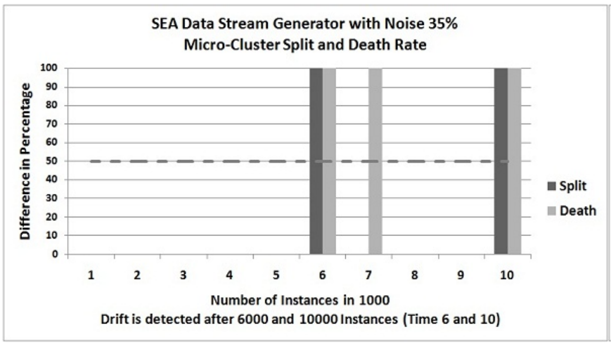
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 2 and 3)
Feature 2	0.00253	0	False Negative
Feature 3	**0.00227**	**1**	True Positive
Feature 1	0.00216	0	

Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 2 and 3)
Feature 3	**0.00121**	**1**	True Positive
Feature 1	0.00021	0	
Feature 2	0.00016	0	False Negative

Figure 5.21: The results of *SEA* data stream generator with a noise level of 25% using Micro-Clusters for tracking features.

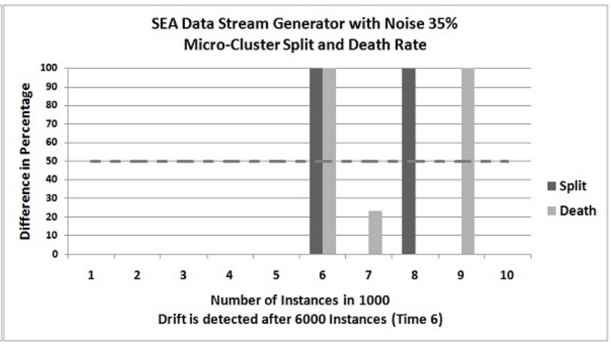
(a) Previous MC-NN
(LPF and **Variance**)



Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 2 and 3)
Feature 2	0.00301	0	False Negative
Feature 3	**0.00267**	**1**	True Positive
Feature 1	0.00229	0	

(b) New MC-NN
(LPF and **IQR**)

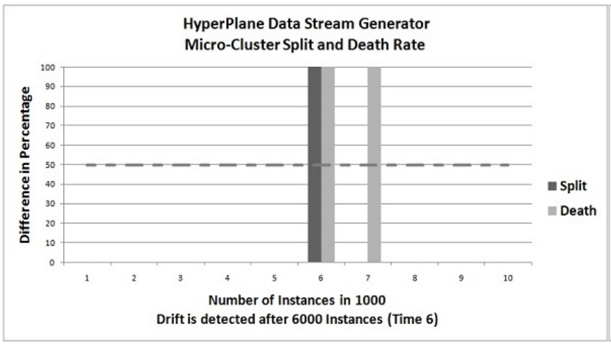


Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 2 and 3)
Feature 2	**0.00731**	**21**	True Positive
Feature 3	**0.00629**	**7**	True Positive
Feature 1	0.00483	10	

Figure 5.22: The results of *SEA* data stream generator with a noise level of 35% using Micro-Clusters for tracking features.

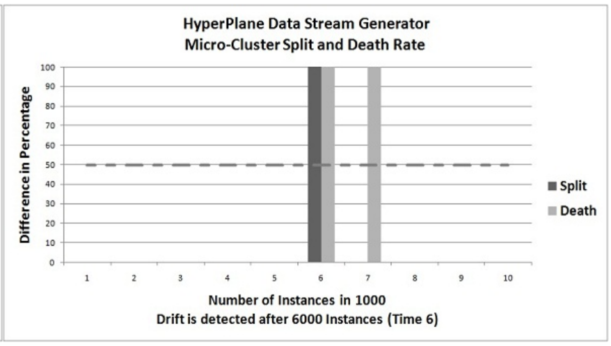
(a) Previous MC-NN
(LPF and **Variance**)



Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 1 and 2)
Feature 3	0.00046	1	False Positive
Feature 2	0.00035	0	False Negative
Feature 1	0.00033	0	False Negative

(b) New MC-NN
(LPF and **IQR**)

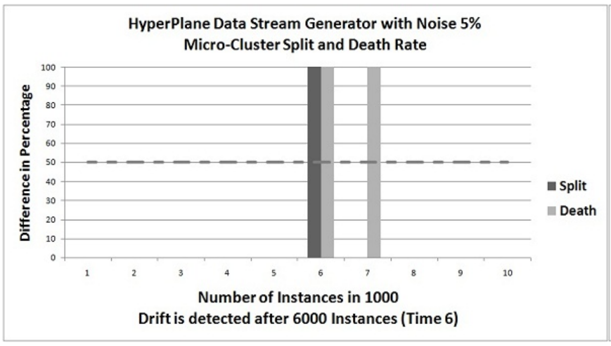


Features are ordered based on *Velocity* rate

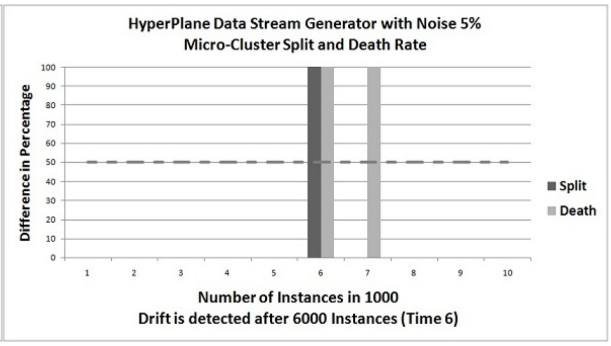
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 1 and 2)
Feature 1	**0.00016**	**1**	True Positive
Feature 2	0.00009	0	False Negative
Feature 3	0.00008	1	

Figure 5.23: The results of *HyperPlane* data stream generator with a noise level of 0% using Micro-Clusters for tracking features.

(a) Previous MC-NN
(LPF and **Variance**)



(b) New MC-NN
(LPF and **IQR**)



Features are ordered based on *Velocity* rate

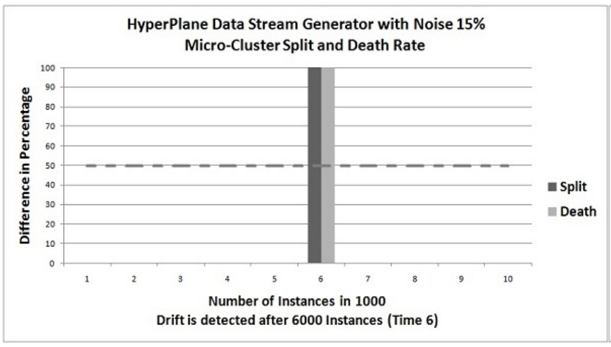
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 1 and 2)
Feature 3	0.00046	1	False Positive
Feature 2	0.00036	0	False Negative
Feature 1	0.00034	0	False Negative

Features are ordered based on *Velocity* rate

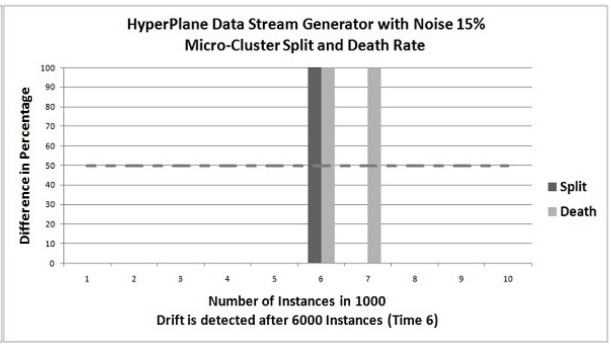
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 1 and 2)
Feature 1	**0.00016**	**1**	True Positive
Feature 2	0.00009	0	False Negative
Feature 3	0.00008	1	

Figure 5.24: The results of *HyperPlane* data stream generator with a noise level of 5% using Micro-Clusters for tracking features.

(a) Previous MC-NN
(LPF and **Variance**)



(b) New MC-NN
(LPF and **IQR**)



Features are ordered based on *Velocity* rate

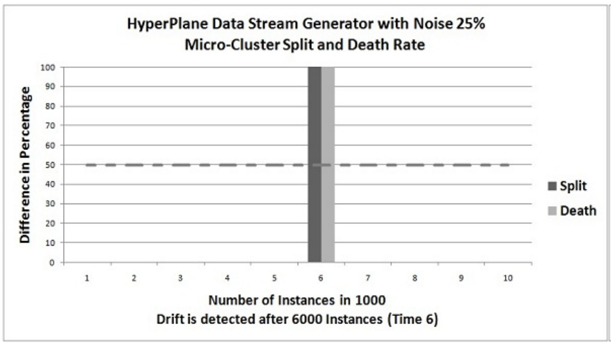
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 1 and 2)
Feature 2	0.00126	0	False Negative
Feature 3	0.00117	2	False Positive
Feature 1	0.00117	0	False Negative

Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 1 and 2)
Feature 1	**0.00019**	**1**	True Positive
Feature 3	0.00014	0	
Feature 2	0.00013	1	False Negative

Figure 5.25: The results of *HyperPlane* data stream generator with a noise level of 15% using Micro-Clusters for tracking features.

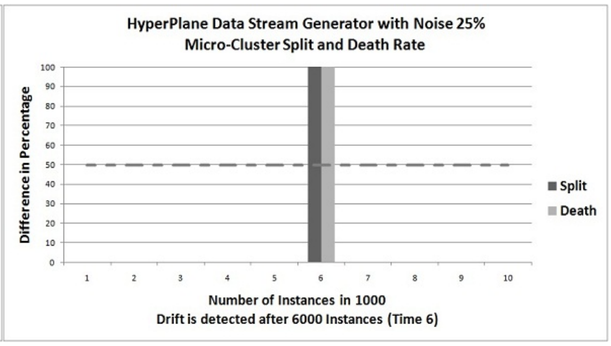
(a) Previous MC-NN
(LPF and **Variance**)



Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 1 and 2)
Feature 2	0.00106	0	False Negative
Feature 1	0.00101	0	False Negative
Feature 3	0.00087	1	

(b) New MC-NN
(LPF and **IQR**)

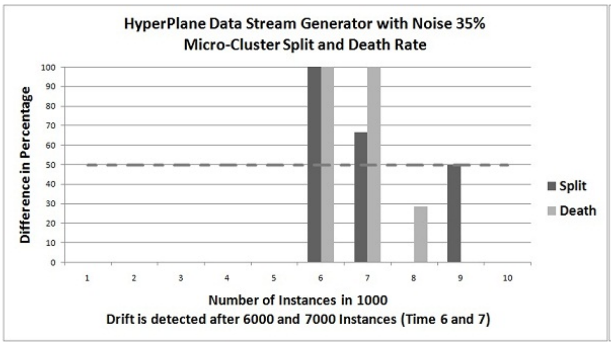


Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 1 and 2)
Feature 3	0.00041	4	False Positive
Feature 1	**0.00039**	**2**	True Positive
Feature 2	0.00027	0	False Negative

Figure 5.26: The results of *HyperPlane* data stream generator with a noise level of 25% using Micro-Clusters for tracking features.

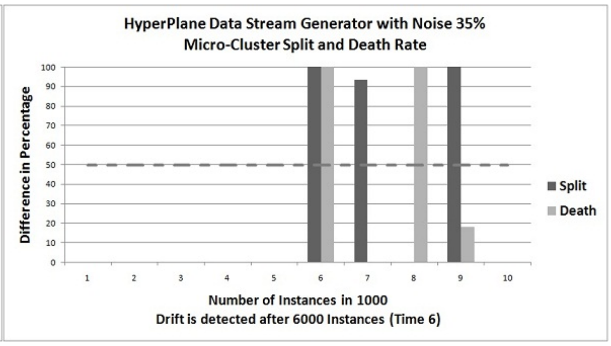
(a) Previous MC-NN
(LPF and **Variance**)



Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 1 and 2)
Feature 3	0.00140	2	False Positive
Feature 2	0.00139	0	False Negative
Feature 1	0.00136	0	False Negative

(b) New MC-NN
(LPF and **IQR**)

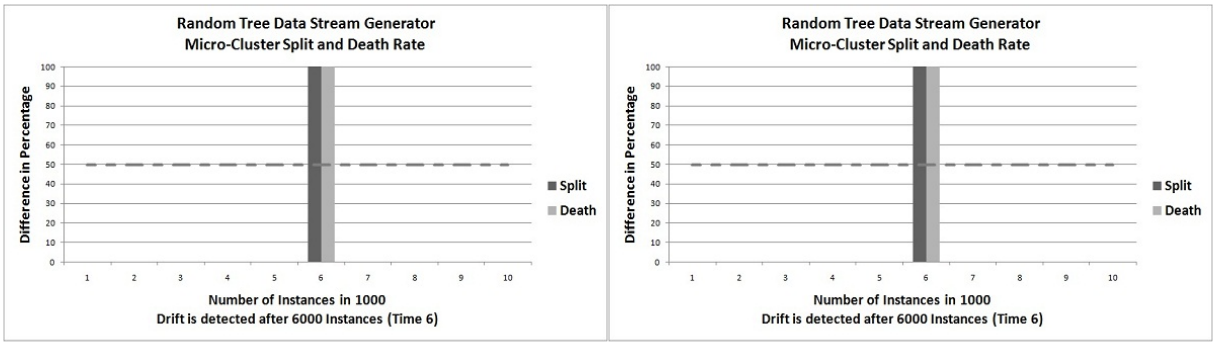


Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 1 and 2)
Feature 1	**0.00077**	**19**	True Positive
Feature 3	0.00076	37	False Positive
Feature 2	0.00069	17	False Negative

Figure 5.27: The results of *HyperPlane* data stream generator with a noise level of 35% using Micro-Clusters for tracking features.

(a) Previous MC-NN (LPF and **Variance**) (b) New MC-NN (LPF and **IQR**)



Features are ordered based on *Velocity* rate

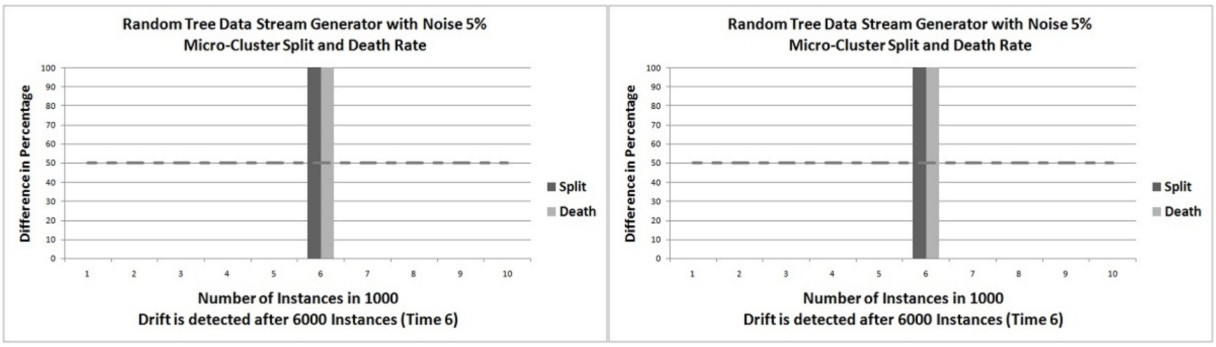
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 1 and 2)
Feature 2	0.00101	0	False Negative
Feature 3	0.00093	0	
Feature 1	0.00090	1	False Negative

Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 1 and 2)
Feature 1	**0.00067**	**17**	True Positive
Feature 2	**0.00066**	**23**	True Positive
Feature 3	0.00063	13	

Figure 5.28: The results of *Random Tree* data stream generator with a noise level of 0% using Micro-Clusters for tracking features.

(a) Previous MC-NN (LPF and **Variance**) (b) New MC-NN (LPF and **IQR**)



Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 1 and 2)
Feature 3	0.00066	0	
Feature 1	**0.00041**	**1**	True Positive
Feature 2	0.00036	0	False Negative

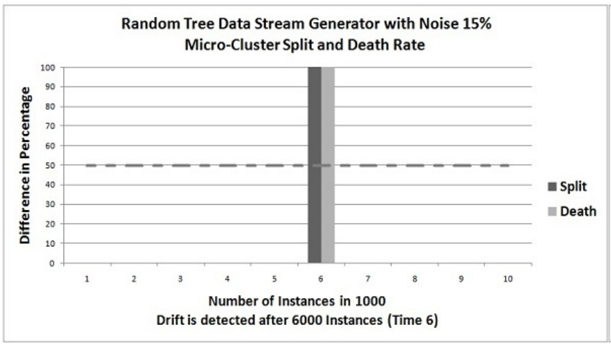
Features are ordered based on *Velocity* rate

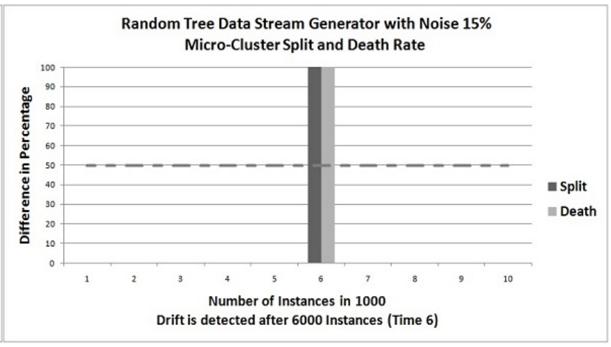
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 1 and 2)
Feature 2	**0.00021**	**6**	True Positive
Feature 3	0.00021	0	
Feature 1	0.00014	1	False Negative

Figure 5.29: The results of *Random Tree* data stream generator with a noise level of 5% using Micro-Clusters for tracking features.

(a) Previous MC-NN
(LPF and **Variance**)

(b) New MC-NN
(LPF and **IQR**)





Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 1 and 2)
Feature 3	0.00142	0	
Feature 1	**0.00138**	**1**	True Positive
Feature 2	0.00115	0	False Negative

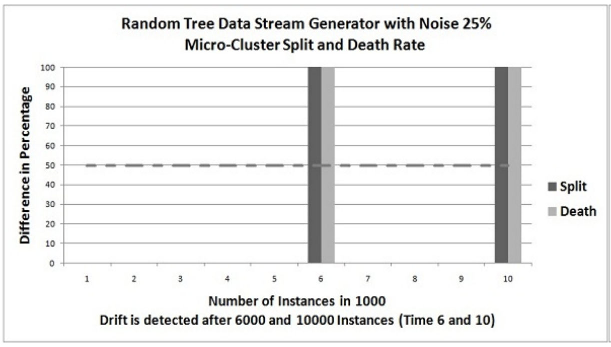
Features are ordered based on *Velocity* rate

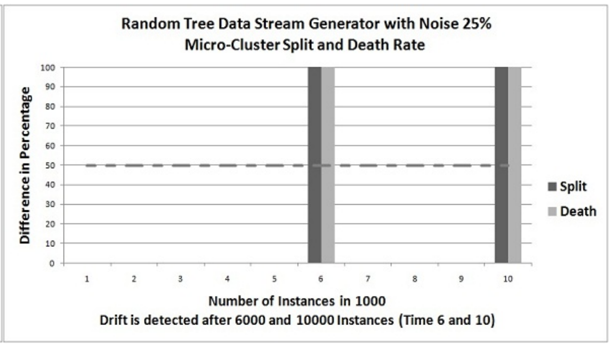
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 1 and 2)
Feature 1	**0.00061**	7	True Positive
Feature 2	**0.00061**	2	True Positive
Feature 3	0.00060	1	

Figure 5.30: The results of *Random Tree* data stream generator with a noise level of 15% using Micro-Clusters for tracking features.

(a) Previous MC-NN
(LPF and **Variance**)

(b) New MC-NN
(LPF and **IQR**)





Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 1 and 2)
Feature 3	0.00165	0	
Feature 2	0.00124	0	False Negative
Feature 1	0.00099	2	False Negative

Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 1 and 2)
Feature 3	0.00068	3	False Positive
Feature 2	**0.00067**	**10**	True Positive
Feature 1	0.00053	2	False Negative

Features are ordered based on *Velocity* rate

Feature	(Time 10) Velocity Rate (Descending Order)	(Time 10) History of Maximum Variance	Detection of Swapped Features (Feature 1 and 2)
Feature 3	0.00071	0	
Feature 2	0.00054	1	False Positive*
Feature 1	0.00044	0	False Negative

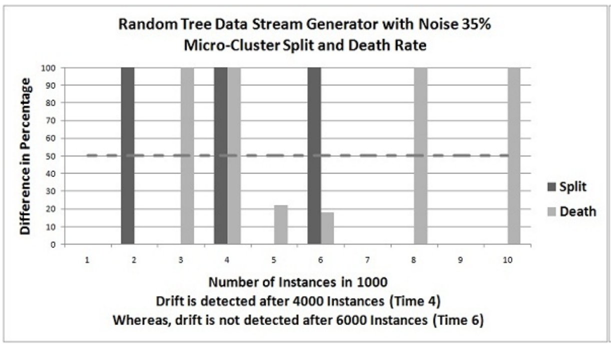
Features are ordered based on *Velocity* rate

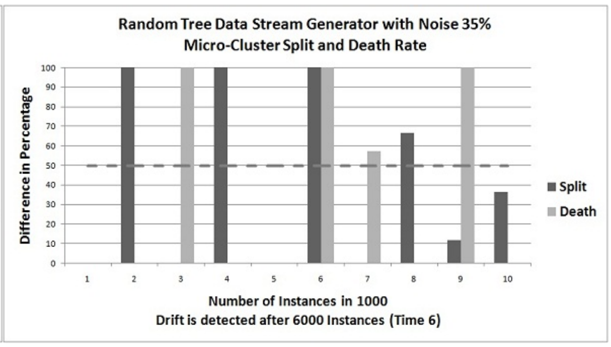
Feature	(Time 10) Velocity Rate (Descending Order)	(Time 10) History of Maximum IQR	Detection of Swapped Features (Feature 1 and 2)
Feature 1	0.00011	19	False Positive*
Feature 2	0.00008	1	False Positive*
Feature 3	0.00008	0	

Figure 5.31: The results of *Random Tree* data stream generator with a noise level of 25% using Micro-Clusters for tracking features.

(a) Previous MC-NN
(LPF and **Variance**)

(b) New MC-NN
(LPF and **IQR**)





Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 1 and 2)
Feature 1	0.00098	1	False Positive(-)
Feature 2	0.00074	0	False Negative
Feature 3	0.00064	2	

Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 1 and 2)
Feature 3	0.00025	2	False Positive
Feature 2	**0.00018**	**2**	True Positive
Feature 1	0.00015	1	False Negative

Figure 5.32: The results of *Random Tree* data stream generator with a noise level of 35% using Micro-Clusters for tracking features.

of maximum *Variance* for concept drift detection and feature tracking, and part (b) of the figures shows the corresponding results for using MC-NN with *IQR* and history of maximum *IQR*. Different figures correspond to different data streams and noise levels. Regarding concept drift detection, the same results as the previous section (Section 4.3) have been achieved for both, using *Variance* and *IQR*. All concept drifts were detected correctly, with only one exception, a noise level 35% with the method based on *Variance* (see Figure 5.32), however, both methods also detected a false concept drift for a noise level 25% and 35%. This is likely not to cause a degradation of classification accuracy, as features are merely flagged up to the feature selection method to have potentially changed their relevance but it is then up to the feature selection method to evaluate these features and decide if they should be included.

If a concept drift is detected, then the methods use the feature *Velocity* and history of maximum *Variance* or *IQR* respectively to decide which features should be flagged up to be considered for inclusion or removal from the currently considered features. The results for this are depicted in the bottom part of Figures 5.18 to 5.32. As a reminder, the features are ranked according to their velocities and the 50% features with the highest *Velocity* are examined closer. Where 50% yielded good results as examined in Appendix E. As there are 3 features in each stream, the 2 features with the highest *Velocity* are always selected. However, if there are 10 features in each stream, as an example, the 5 features with the highest *Velocity* are always selected. These features are then flagged up if they also appeared in the history of maximum *Variance* or

IQR in the previous time window. This has already been explained in greater detail in Section 6.1. It is expected here that features that have been swapped during the concept drift are more likely to be flagged up for inclusion or removal from the currently considered set of features.

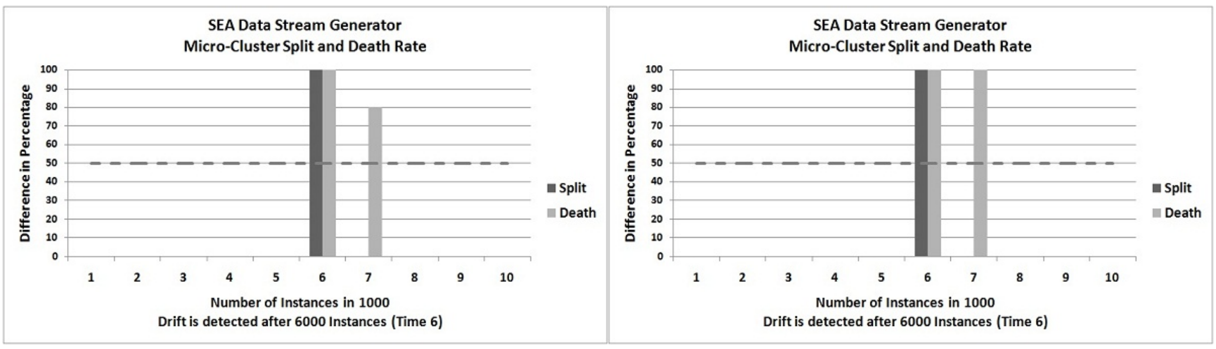
Table 5.1: Summary of the experimental results with artificial datasets generated with noise levels of 0%, 5%, 15%, 25%, and 35%. The results are reported for the Time 6 which is the time of swapped features.

Generator	True Positive (*Variance*)	True Positive (*IQR*)	False Positive (*Variance*)	False Positive (*IQR*)
SEA with Noise 0%	2	2		
SEA with Noise 5%	1	2		
SEA with Noise 15%	1	1		
SEA with Noise 25%	1	1		
SEA with Noise 35%	1	2		
HyperPlane with Noise 0%		1	1	
HyperPlane with Noise 5%		1	1	
HyperPlane with Noise 15%		1	1	
HyperPlane with Noise 25%		1		1
HyperPlane with Noise 35%		1	1	1
Random Tree with Noise 0%		2		
Random Tree with Noise 5%	1	1		
Random Tree with Noise 15%	1	2		
Random Tree with Noise 25%		1	1*	1 + 2*
Random Tree with Noise 35%		1	1(-)	1
Total:	8	20	6	6

Table 5.1 summarises the feature tracking results presented in Figures 5.18 to 5.32. Numbers indicated with a '*' in Table 5.1 indicate *false positives* detections of features that changed their relevance. This means that there was also an unexpected concept drift. However, it is not possible to verify with absolute certainty if the detection was indeed a *false positive*. Whereas, numbers indicated with a (-) in Table 5.1 reflect that a known drift was not detected. It can be seen that the here presented method based on maximum *IQR* achieves a much higher *true positive* detection of features that changed. It also has a lower number of *false positives* detection with no '*' (4 *false positives*) compared with the method based on *Variance* (5 *false positives*). Now the *true positive* detections of features are based on the fact the features that are known to have changed as they have been swapped at the time of concept drift. However, considering the fact that each artificial stream generator's native method for inducing a concept drift has been used as well some of the *false positive* detections may very well be *true positives*. Even if they were not listed in Table 5.1 as *true positives*, it would merely mean that they are flagged to the feature selection method to be reconsidered for inclusion or exclusion of the feature set to be considered for adaptation. Loosely speaking, correct *true positive* detection numbers are more

(a) Previous MC-NN (LPF and **Variance**)

(b) New MC-NN (LPF and **IQR**)



Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 2 and 3)
Feature 3	**0.00772**	**2**	True Positive
Feature 2	0.00754	0	False Negative
Feature 1	0.00721	0	

Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 2 and 3)
Feature 3	**0.00180**	**5**	True Positive
Feature 2	**0.00097**	**1**	True Positive
Feature 1	0.00071	0	

Figure 5.33: The results of *SEA* data stream generator with a noise level of 0% using Micro-Clusters for tracking features.

important than *false positive* detection numbers.

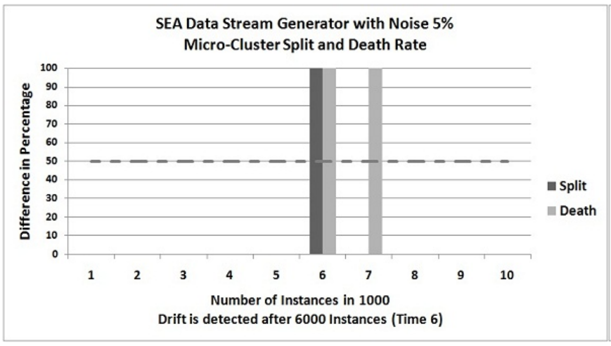
In order to remove the possibility of changing feature relevance due to artificial concept drift generators, the feature tracking experiments on the artificial datasets were re-conducted. This time no concept drift was introduced apart from the swapping of the features. Thus the only features that have changed their relevance must be the swapped features. The results of these experiments are displayed in Figures 5.33 to 5.47. As it can be seen, and as expected, both methods have a lower *false positive* count of flagging features that changed relevance in comparison with the results presented in Figures 5.18 to 5.32. Again the developed method based on *IQR* performs best, it detected most of the features that were swapped.

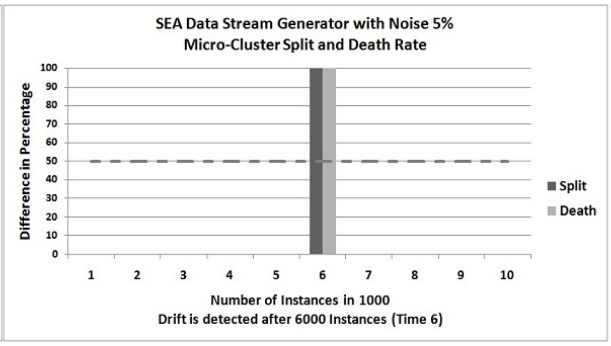
Table 5.2 summarises these experiments. Altogether there were 30 features swapped so they all changed their relevance and thus should be identified by the methods. As it can be seen the method based on original MC-NN with *Variance* identifies 6 features correctly and also resulted in 3 *false positives*. The developed method based on *IQR* detects 26 features that changed their relevance correctly and resulted in 2 *false positives*.

Table 5.2 summarises the feature tracking results presented in Figures 5.33 to 5.47. Numbers indicated with a '*' in Table 5.2 indicate *false positive* detection of features that changed their relevance. It can be seen that the developed method based on *IQR* achieves a much higher *true positive* detection of features that changed with only 2 *false positives*. The *true positives* detection of features are based on the fact the features that are known to have changed as they

(a) Previous MC-NN
(LPF and **Variance**)

(b) New MC-NN
(LPF and **IQR**)





Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 2 and 3)
Feature 2	0.00288	0	False Negative
Feature 3	**0.00211**	**1**	True Positive
Feature 1	0.00202	0	

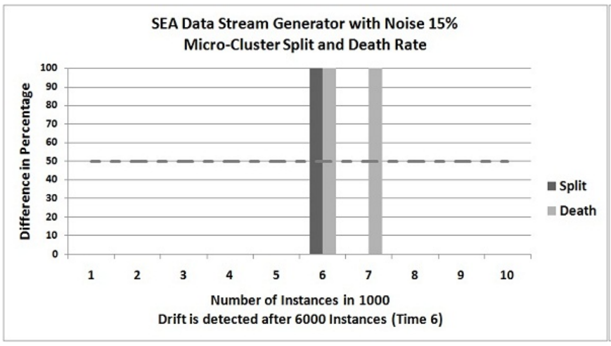
Features are ordered based on *Velocity* rate

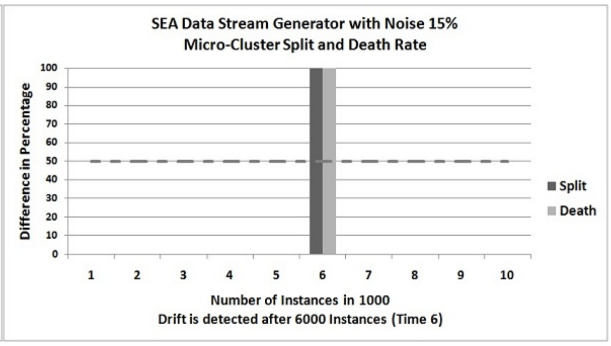
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 2 and 3)
Feature 3	**0.00377**	**7**	True Positive
Feature 2	**0.00376**	**18**	True Positive
Feature 1	0.00326	1	

Figure 5.34: The results of *SEA* data stream generator with a noise level of 5% using Micro-Clusters for tracking features.

(a) Previous MC-NN
(LPF and **Variance**)

(b) New MC-NN
(LPF and **IQR**)





Features are ordered based on *Velocity* rate

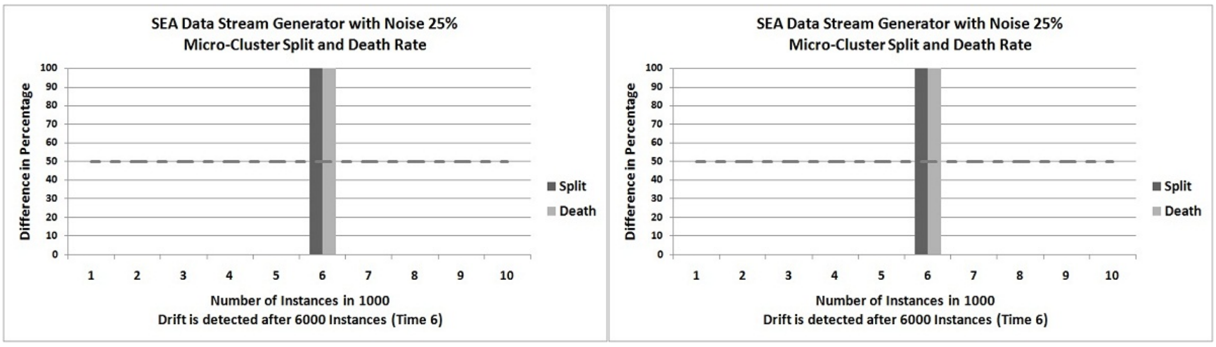
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 2 and 3)
Feature 2	0.00278	0	False Negative
Feature 3	**0.00236**	**1**	True Positive
Feature 1	0.00223	0	

Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 2 and 3)
Feature 2	**0.00301**	**11**	True Positive
Feature 3	**0.00281**	**21**	True Positive
Feature 1	0.00280	3	

Figure 5.35: The results of *SEA* data stream generator with a noise level of 15% using Micro-Clusters for tracking features.

(a) Previous MC-NN (LPF and **Variance**) (b) New MC-NN (LPF and **IQR**)



Features are ordered based on *Velocity* rate

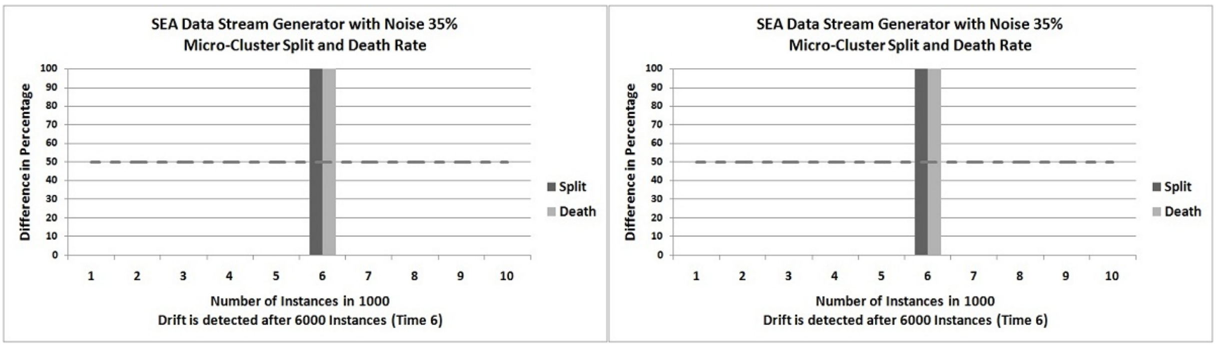
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 2 and 3)
Feature 1	0.01186	0	
Feature 2	0.01090	0	False Negative
Feature 3	0.01017	2	False Negative

Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 2 and 3)
Feature 2	**0.00371**	7	True Positive
Feature 3	**0.00342**	**4**	True Positive
Feature 1	0.00334	1	

Figure 5.36: The results of *SEA* data stream generator with a noise level of 25% using Micro-Clusters for tracking features.

(a) Previous MC-NN (LPF and **Variance**) (b) New MC-NN (LPF and **IQR**)



Features are ordered based on *Velocity* rate

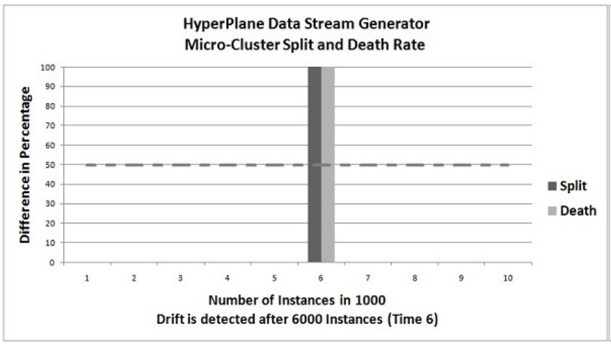
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 2 and 3)
Feature 3	0.00747	0	False Negative
Feature 1	0.00710	0	
Feature 2	0.00696	1	False Negative

Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 2 and 3)
Feature 1	0.00273	0	
Feature 2	**0.00261**	**8**	True Positive
Feature 3	0.00246	6	False Negative

Figure 5.37: The results of *SEA* data stream generator with a noise level of 35% using Micro-Clusters for tracking features.

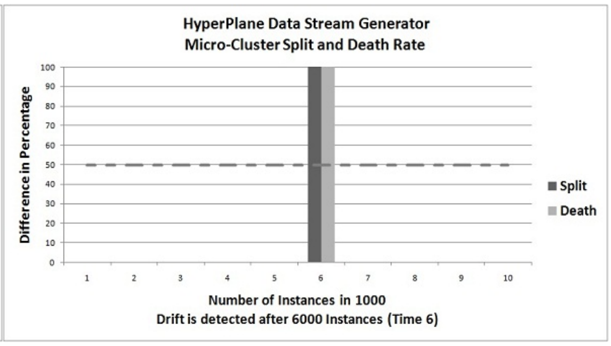
(a) Previous MC-NN
(LPF and **Variance**)



Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 1 and 2)
Feature 3	0.00092	1	False Positive
Feature 1	**0.00086**	**1**	True Positive
Feature 2	0.00070	0	False Negative

(b) New MC-NN
(LPF and **IQR**)

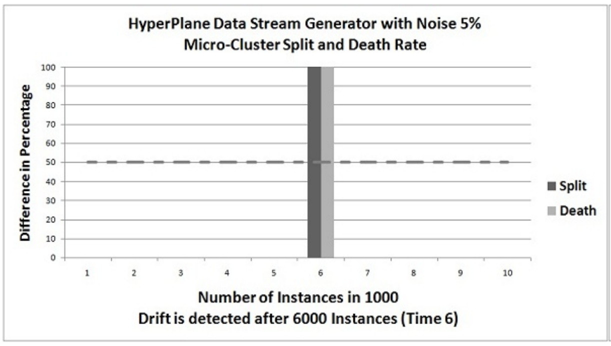


Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 1 and 2)
Feature 1	**0.00013**	**2**	True Positive
Feature 2	0.00009	0	False Negative
Feature 3	0.00006	1	

Figure 5.38: The results of *HyperPlane* data stream generator with a noise level of 0% using Micro-Clusters for tracking features.

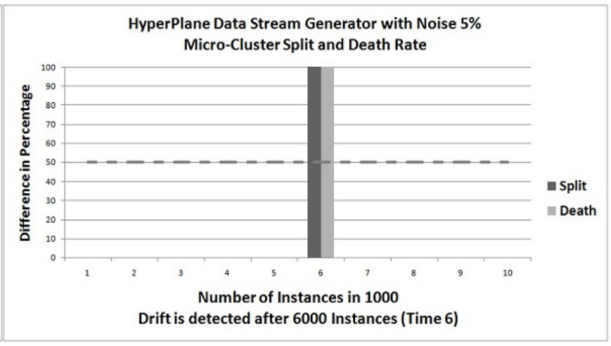
(a) Previous MC-NN
(LPF and **Variance**)



Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 1 and 2)
Feature 2	**0.00248**	**2**	True Positive
Feature 1	0.00215	0	False Negative
Feature 3	0.00196	1	

(b) New MC-NN
(LPF and **IQR**)



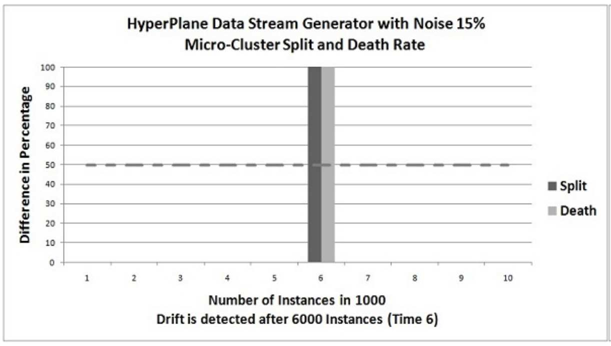
Features are ordered based on *Velocity* rate

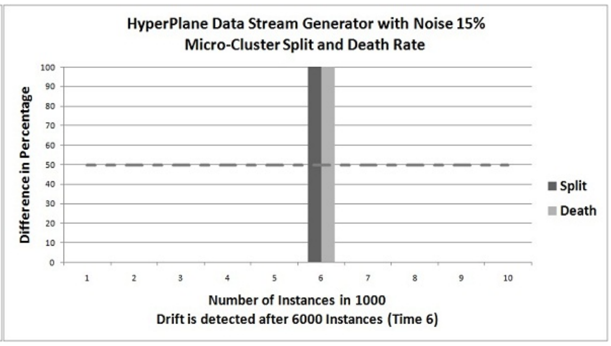
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 1 and 2)
Feature 1	**0.00024**	**3**	True Positive
Feature 3	0.00022	0	
Feature 2	0.00013	1	False Negative

Figure 5.39: The results of *HyperPlane* data stream generator with a noise level of 5% using Micro-Clusters for tracking features.

(a) Previous MC-NN
(LPF and **Variance**)

(b) New MC-NN
(LPF and **IQR**)





Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 1 and 2)
Feature 1	0.00190	0	False Negative
Feature 3	0.00190	1	False Positive
Feature 2	0.00178	1	False Negative

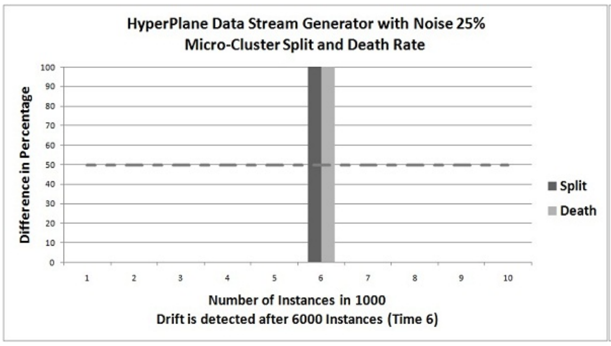
Features are ordered based on *Velocity* rate

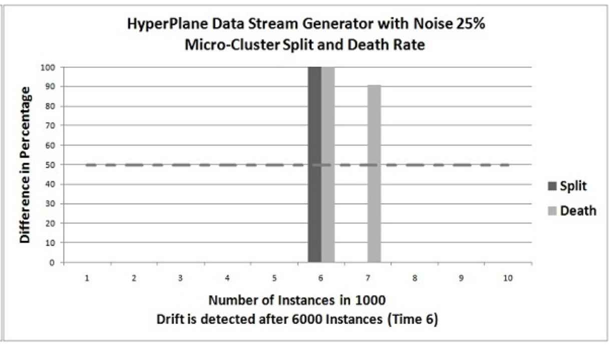
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 1 and 2)
Feature 2	**0.00072**	**8**	True Positive
Feature 1	**0.00071**	**21**	True Positive
Feature 3	0.00070	20	

Figure 5.40: The results of *HyperPlane* data stream generator with a noise level of 15% using Micro-Clusters for tracking features.

(a) Previous MC-NN
(LPF and **Variance**)

(b) New MC-NN
(LPF and **IQR**)





Features are ordered based on *Velocity* rate

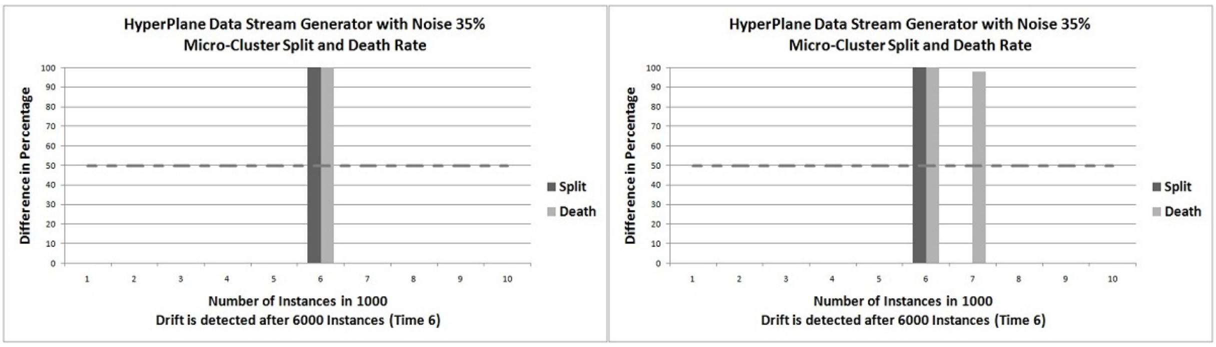
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 1 and 2)
Feature 2	**0.00158**	**1**	True Positive
Feature 1	0.00133	0	False Negative
Feature 3	0.00121	0	

Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 1 and 2)
Feature 2	**0.00029**	**9**	True Positive
Feature 1	**0.00027**	**11**	True Positive
Feature 3	0.00026	10	

Figure 5.41: The results of *HyperPlane* data stream generator with a noise level of 25% using Micro-Clusters for tracking features.

(a) Previous MC-NN (LPF and **Variance**) | (b) New MC-NN (LPF and **IQR**)



Features are ordered based on *Velocity* rate

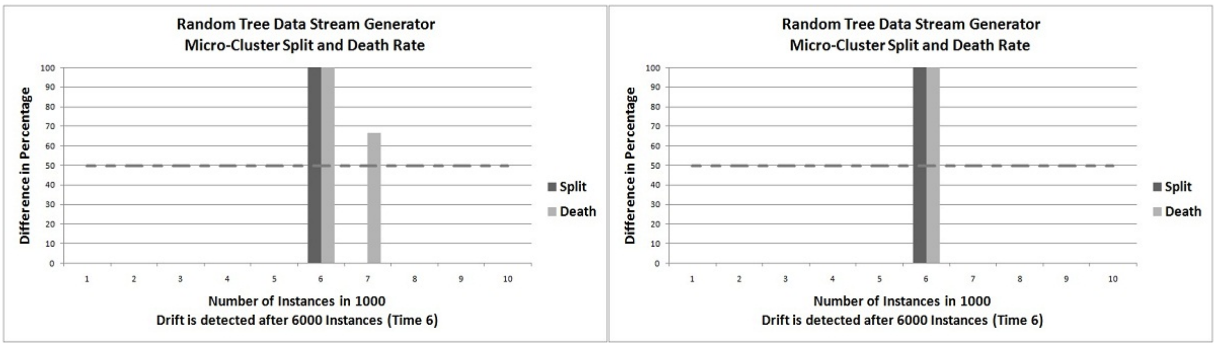
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 1 and 2)
Feature 1	0.00151	0	False Negative
Feature 2	0.00143	0	False Negative
Feature 3	0.00141	0	

Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 1 and 2)
Feature 2	**0.00043**	**87**	True Positive
Feature 1	**0.00042**	**67**	True Positive
Feature 3	0.00040	46	

Figure 5.42: The results of *HyperPlane* data stream generator with a noise level of 35% using Micro-Clusters for tracking features.

(a) Previous MC-NN (LPF and **Variance**) | (b) New MC-NN (LPF and **IQR**)



Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 1 and 2)
Feature 3	0.00050	0	
Feature 2	0.00029	0	False Negative
Feature 1	0.00027	1	False Negative

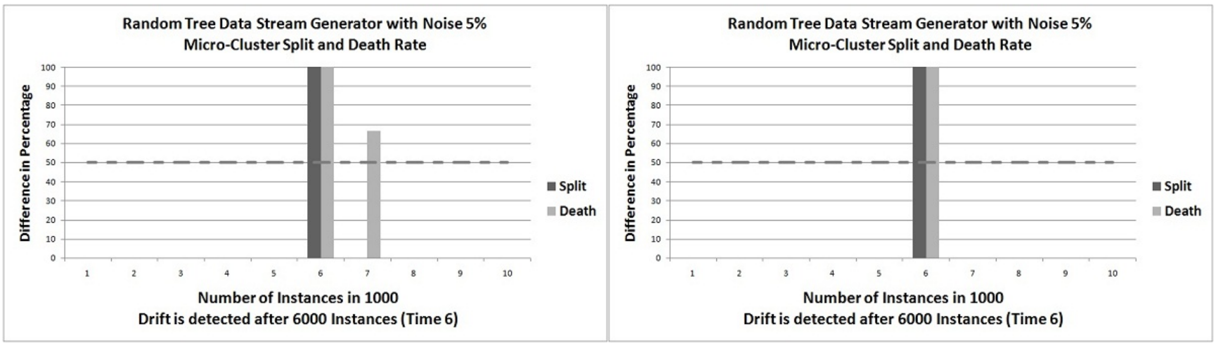
Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 1 and 2)
Feature 1	**0.00030**	**5**	True Positive
Feature 2	**0.00030**	**4**	True Positive
Feature 3	0.00029	2	

Figure 5.43: The results of *Random Tree* data stream generator with a noise level of 0% using Micro-Clusters for tracking features.

(a) Previous MC-NN
(LPF and **Variance**)

(b) New MC-NN
(LPF and **IQR**)



Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 1 and 2)
Feature 3	0.00049	0	
Feature 2	0.00028	0	False Negative
Feature 1	0.00027	1	False Negative

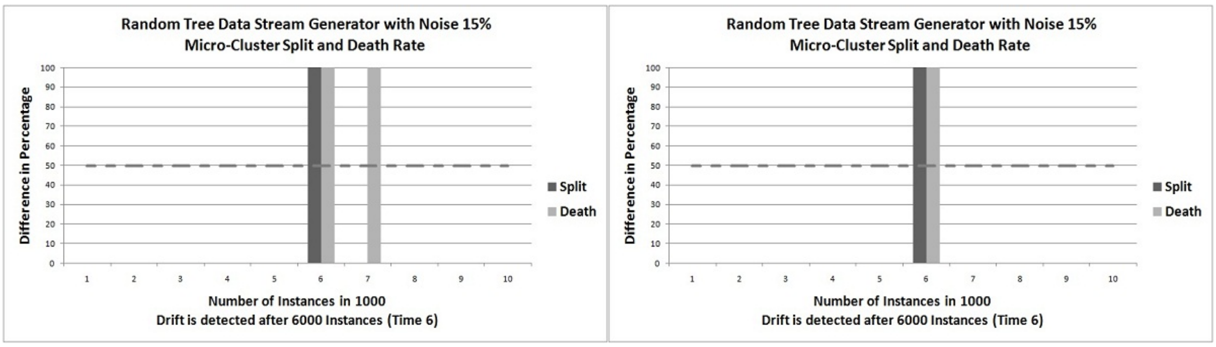
Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 1 and 2)
Feature 1	**0.00030**	**5**	True Positive
Feature 2	**0.00030**	**4**	True Positive
Feature 3	0.00029	2	

Figure 5.44: The results of *Random Tree* data stream generator with a noise level of 5% using Micro-Clusters for tracking features.

(a) Previous MC-NN
(LPF and **Variance**)

(b) New MC-NN
(LPF and **IQR**)



Features are ordered based on *Velocity* rate

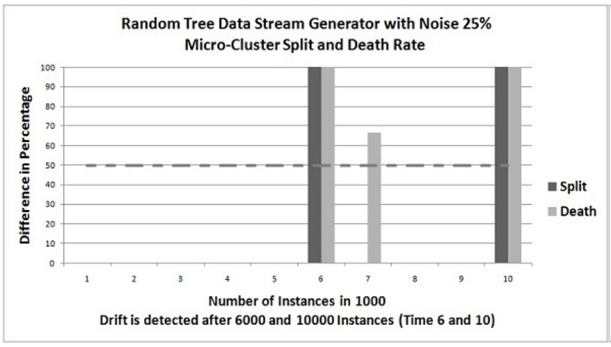
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 1 and 2)
Feature 3	0.00051	0	
Feature 1	0.00042	0	False Negative
Feature 2	0.00041	1	False Negative

Features are ordered based on *Velocity* rate

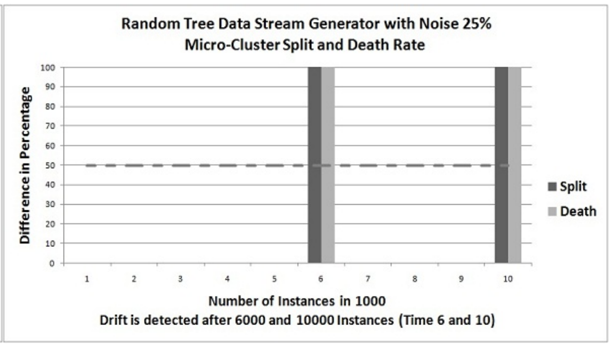
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 1 and 2)
Feature 1	**0.00031**	**20**	True Positive
Feature 2	**0.00023**	**1**	True Positive
Feature 3	0.00019	0	

Figure 5.45: The results of *Random Tree* data stream generator with a noise level of 15% using Micro-Clusters for tracking features.

(a) Previous MC-NN
(LPF and **Variance**)



(b) New MC-NN
(LPF and **IQR**)



Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 1 and 2)
Feature 3	0.00044	0	
Feature 2	0.00032	0	False Negative
Feature 1	0.00031	1	False Negative

Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 1 and 2)
Feature 1	**0.00008**	**2**	True Positive
Feature 3	0.00006	0	
Feature 2	0.00005	0	False Negative

Features are ordered based on *Velocity* rate

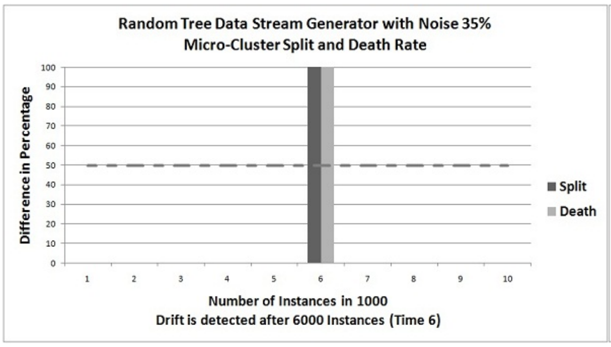
Feature	(Time 10) Velocity Rate (Descending Order)	(Time 10) History of Maximum Variance	Detection of Swapped Features (Feature 1 and 2)
Feature 3	0.00124	0	
Feature 2	0.00116	1	False Positive*
Feature 1	0.00079	0	False Negative

Features are ordered based on *Velocity* rate

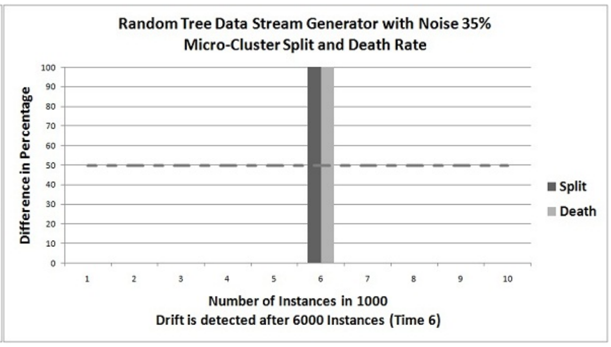
Feature	(Time 10) Velocity Rate (Descending Order)	(Time 10) History of Maximum IQR	Detection of Swapped Features (Feature 1 and 2)
Feature 2	0.00068	4	False Positive*
Feature 1	0.00065	4	False Positive*
Feature 3	0.00056	0	

Figure 5.46: The results of *Random Tree* data stream generator with a noise level of 25% using Micro-Clusters for tracking features.

(a) Previous MC-NN
(LPF and **Variance**)



(b) New MC-NN
(LPF and **IQR**)



Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 1 and 2)
Feature 1	0.00223	0	False Negative
Feature 2	0.00195	0	False Negative
Feature 3	0.00177	1	

Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 1 and 2)
Feature 2	**0.00070**	**1**	True Positive
Feature 1	**0.00055**	**3**	True Positive
Feature 3	0.00054	6	

Figure 5.47: The results of *Random Tree* data stream generator with a noise level of 35% using Micro-Clusters for tracking features.

Table 5.2: Summary of the experimental results with artificial datasets generated with noise levels of 0%, 5%, 15%, 25%, and 35%. The results are reported for Time 6 which is the point at which features had been swapped.

Generator	True Positive (*Variance*)	True Positive (*IQR*)	False Positive (*Variance*)	False Positive (*IQR*)
SEA with Noise 0%	1	2		
SEA with Noise 5%	1	2		
SEA with Noise 15%	1	2		
SEA with Noise 25%		2		
SEA with Noise 35%		1		
HyperPlane with Noise 0%	1	1	1	
HyperPlane with Noise 5%	1	1		
HyperPlane with Noise 15%		2	1	
HyperPlane with Noise 25%	1	2		
HyperPlane with Noise 35%		2		
Random Tree with Noise 0%		2		
Random Tree with Noise 5%		2		
Random Tree with Noise 15%		2		
Random Tree with Noise 25%		1	1*	2*
Random Tree with Noise 35%		2		
Total:	6	26	3	2

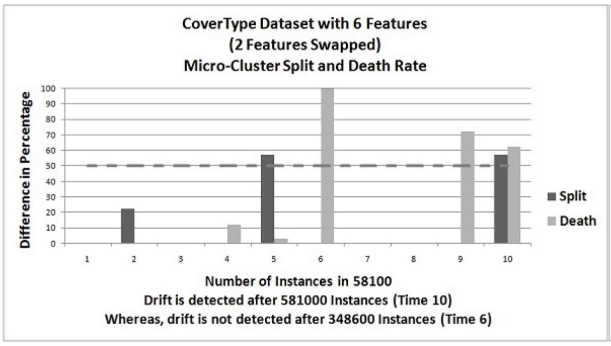
have been swapped at the time of concept drift.

Next, both approaches (i.e., the developed method with *Variance* and *IQR*) were tested on the controlled real datasets described in Table 3.3. For the experiment, the default parameters (stated in Table 3.3) of the method were used unless stated otherwise. Real-time Min-Max *Normalisation* and *LPF* were applied to minimise the effect of feature-bias and noise, respectively. Two versions of each dataset were used with two features (indexes of swapped features are 1 with 4), and four features (indexes of swapped features are 2 and 3 with 5 and 6) have been selected randomly to be swapped in order to validate whether the feature tracking method can identify the changed features correctly.

Figures 5.48 to 5.57 visualise the results for concept drift detection and feature tracking using controlled real datasets. Again part (a) of the figures refers to the method based on *Variance* and part (b) of the figures refers to the method based on *IQR*.

Regarding drift detection in the figures, it can be seen that the *Split* and *Death* rates at the time of concept drift increase (i.e., time when the features were swapped), indicating that the current set of Micro-Clusters does not fit the concept encoded in the data anymore, and a concept drift is detected correctly for every dataset. The method based on *IQR* examined in the figures. Please note there are further concept drifts detected at different times than the features

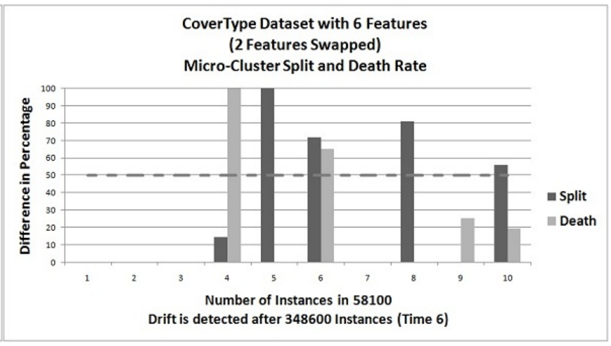
(a) Previous MC-NN
(Real-Time Normalisation and LPF)
(Variance)



Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Features 1 and 4)
Feature 6	0.14926	2	False Positive(-)
Feature 4	0.11770	0	False Negative
Feature 1	0.09163	5	False Positive(-)
Feature 2	0.05656	0	
Feature 5	0.05491	2	
Feature 3	0.04745	0	

(b) New MC-NN
(Real-Time Normalisation and LPF)
(IQR)



Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Features 1 and 4)
Feature 6	0.00114	141	False Positive
Feature 1	**0.00105**	**40**	True Positive
Feature 4	**0.00073**	**1**	True Positive
Feature 2	0.00071	0	
Feature 5	0.00062	19	
Feature 3	0.00053	64	

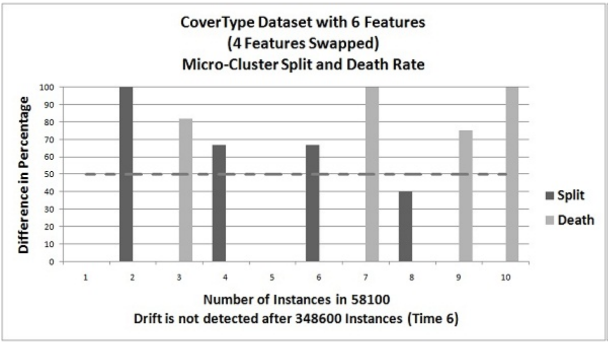
Figure 5.48: The results of *CoverType* dataset with 6 features (2 features swapped) using Micro-Clusters for tracking features.

were swapped, which could be due to the fact that real data was used and thus there may be concept drifts that are unknown. This is one of the reasons why the method was also evaluated on artificial datasets earlier in this chapter, as the ground truth for the artificial datasets was known. At this stage, it is assumed that high *Split* and *Death* rates other than during the time when features were swapped are due to natural concept drift. As the ground truth for the concept drift is known only at the time of feature swapping, the evaluation of correct feature tracking was focussed on the time of feature swapping only.

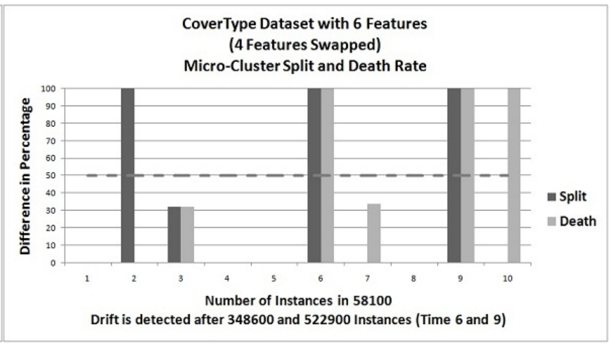
Regarding the results for the datasets based on *CoverType* (see Figures 5.48 and 5.49), *Diabetic Retinopathy Debrecen* (see Figure 5.51), *Gesture Phase Segmentation* (see Figures 5.52 and 5.53), and *Waveform (with Noise)* (see Figures 5.56 and 5.57), the method based on *IQR* is superior by identifying the swapped features (i.e., one of the swapped feature or more) correctly. Whereas, the method based on *Variance* did not identify any correctly, because a drift was not detected. For the *Diabetic Retinopathy Debrecen* dataset in Figure 5.50, the *Statlog (Landsat Satellite)* in Figures 5.54 and 5.55, both methods identify a concept drift correctly at the time the features were swapped. Again the method based on *IQR* was superior to the method based on *Variance*, as it identified the swapped features (i.e., one of the swapped feature or more) correctly. Whereas, the method based on *Variance* did not identify any correctly.

These results are summarised in Table 5.3. It can be seen that the method based on *IQR*

(a) Previous MC-NN
(Real-Time Normalisation and LPF)
(Variance)



(b) New MC-NN
(Real-Time Normalisation and LPF)
(IQR)



Features are ordered based on *Velocity* rate

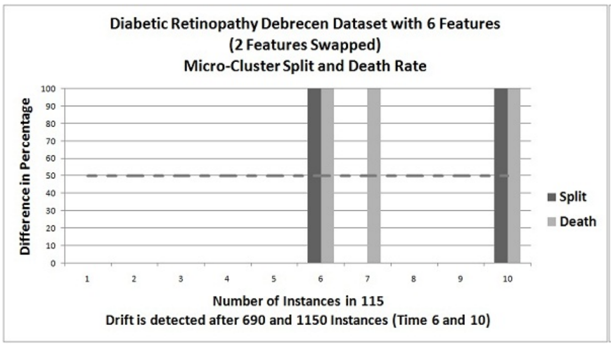
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Features 2,3,5 and 6)
Feature 1	0.00104	1	False Positive(-)
Feature 6	0.00058	0	False Negative
Feature 2	0.00036	0	False Negative
Feature 4	0.00033	0	
Feature 5	0.00031	1	False Negative
Feature 3	0.00014	0	False Negative

Features are ordered based on *Velocity* rate

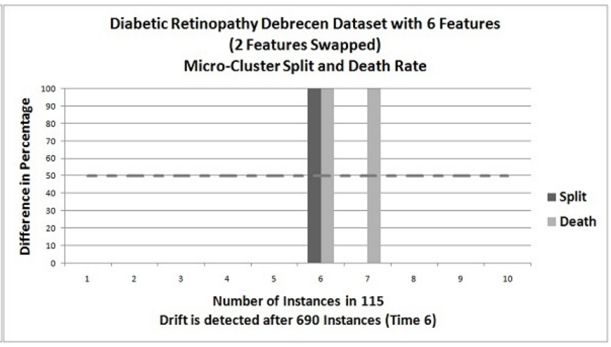
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Features 2,3,5 and 6)
Feature 6	**0.00034**	**12**	True Positive
Feature 1	0.00024	63	False Positive
Feature 2	**0.00013**	**14**	True Positive
Feature 5	0.0010	0	False Negative
Feature 4	0.00009	0	
Feature 3	0.00005	0	False Negative

Figure 5.49: The results of *CoverType* dataset with 6 features (4 features swapped) using Micro-Clusters for tracking features.

(a) Previous MC-NN
(Real-Time Normalisation and LPF)
(Variance)



(b) New MC-NN
(Real-Time Normalisation and LPF)
(IQR)



Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Features 1 and 4)
Feature 5	0.68994	0	
Feature 6	0.43132	0	
Feature 2	0.30771	0	
Feature 1	0.27583	1	False Negative
Feature 3	0.20650	0	
Feature 4	0.18613	0	False Negative

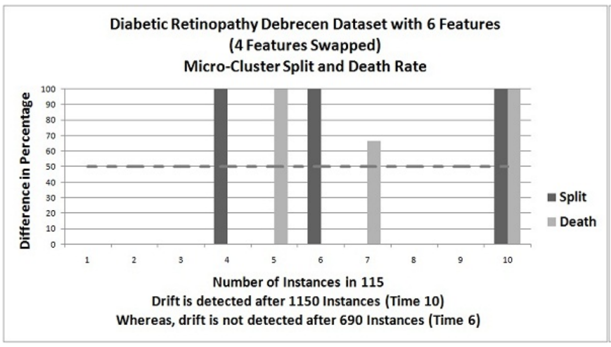
Features are ordered based on *Velocity* rate

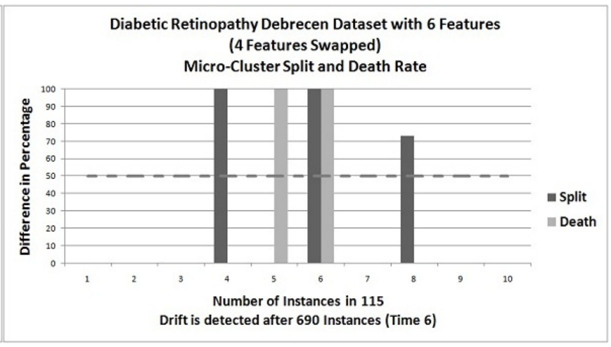
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Features 1 and 4)
Feature 1	**0.03930**	**12**	True Positive
Feature 3	0.02474	0	
Feature 2	0.02404	4	False Positive
Feature 6	0.01958	0	
Feature 5	0.01651	0	
Feature 4	0.00062	0	False Negative

Figure 5.50: The results of *Diabetic Retinopathy Debrecen* dataset with 6 features (2 features swapped) using Micro-Clusters for tracking features.

(a) Previous MC-NN
(Real-Time Normalisation and LPF)
(Variance)

(b) New MC-NN
(Real-Time Normalisation and LPF)
(IQR)





Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Features 2,3,5 and 6)
Feature 6	0.30681	0	False Negative
Feature 5	0.30026	1	False Positive(-)
Feature 1	0.18628	0	
Feature 3	0.05738	0	False Negative
Feature 2	0.04166	0	False Negative
Feature 4	0.00020	0	

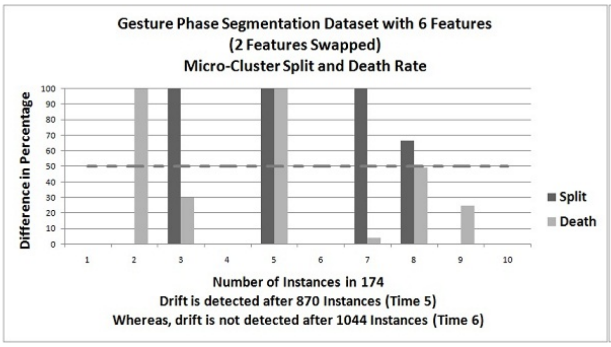
Features are ordered based on *Velocity* rate

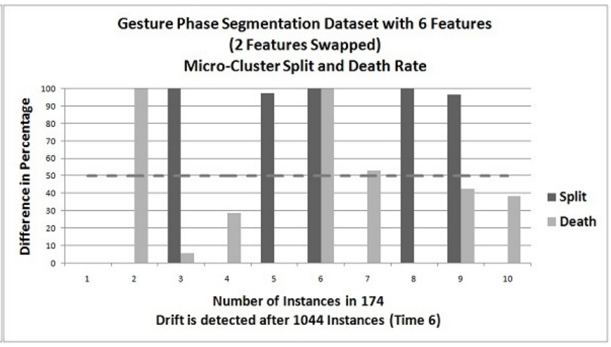
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Features 2,3,5 and 6)
Feature 5	**0.05930**	**15**	True Positive
Feature 6	**0.05743**	**26**	True Positive
Feature 1	0.03030	1	False Positive
Feature 2	0.01644	0	False Negative
Feature 3	0.01327	0	False Negative
Feature 4	0.00003	0	

Figure 5.51: The results of *Diabetic Retinopathy Debrecen* dataset with 6 features (4 features swapped) using Micro-Clusters for tracking features.

(a) Previous MC-NN
(Real-Time Normalisation and LPF)
(Variance)

(b) New MC-NN
(Real-Time Normalisation and LPF)
(IQR)





Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Features 1 and 4)
Feature 4	1.89728	0	False Negative
Feature 6	1.21553	0	
Feature 5	0.98148	0	
Feature 3	0.93844	0	
Feature 2	0.50607	0	
Feature 1	0.01458	0	False Negative

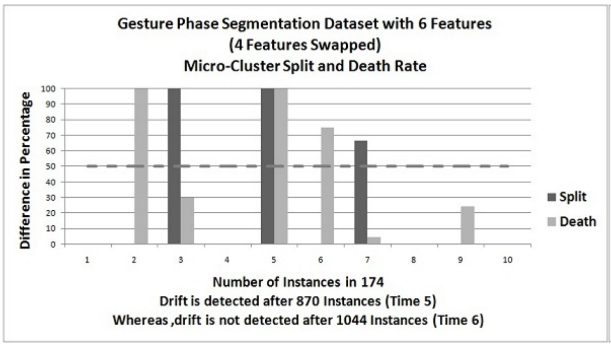
Features are ordered based on *Velocity* rate

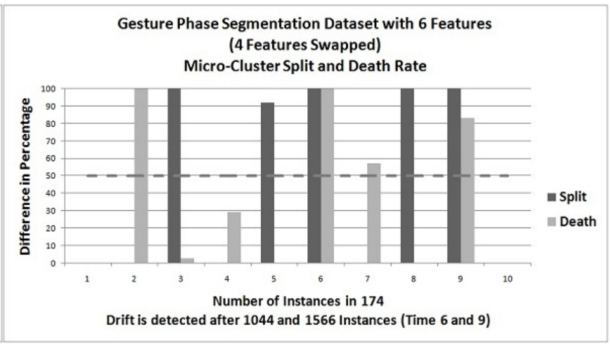
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Features 1 and 4)
Feature 1	**0.00985**	**95**	True Positive
Feature 2	0.00358	0	
Feature 4	**0.00322**	**3**	True Positive
Feature 3	0.00319	1	
Feature 5	0.00103	85	
Feature 6	0.00023	0	

Figure 5.52: The results of *Gesture Phase Segmentation* dataset with 6 features (2 features swapped) using Micro-Clusters for tracking features.

(a) Previous MC-NN
(Real-Time Normalisation and LPF)
(Variance)

(b) New MC-NN
(Real-Time Normalisation and LPF)
(IQR)





Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Features 2,3,5 and 6)
Feature 3	3.86580	0	False Negative
Feature 5	3.64346	1	False Positive(-)
Feature 1	3.19028	0	
Feature 4	2.30119	0	
Feature 6	0.02770	0	False Negative
Feature 2	0.01711	0	False Negative

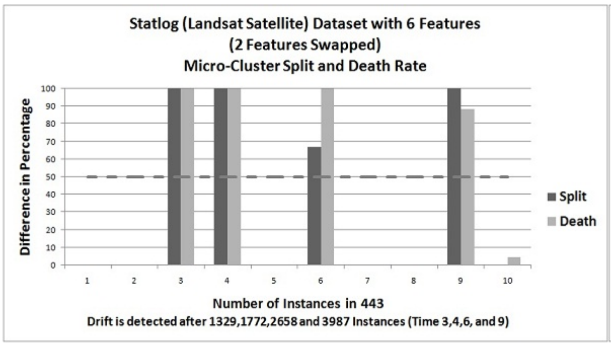
Features are ordered based on *Velocity* rate

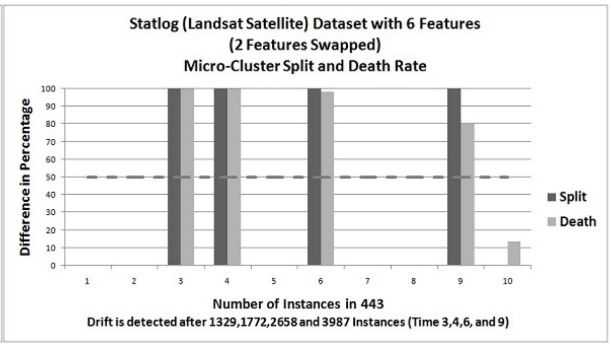
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Features 2,3,5 and 6)
Feature 2	**0.01193**	**96**	True Positive
Feature 6	0.00960	0	False Negative
Feature 3	**0.00232**	**1**	True Positive
Feature 4	0.00089	87	
Feature 1	0.00051	0	
Feature 5	0.00031	0	False Negative

Figure 5.53: The results of *Gesture Phase Segmentation* dataset with 6 features (4 features swapped) using Micro-Clusters for tracking features.

(a) Previous MC-NN
(Real-Time Normalisation and LPF)
(Variance)

(b) New MC-NN
(Real-Time Normalisation and LPF)
(IQR)





Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Features 1 and 4)
Feature 1	**1.51620**	**1**	True Positive
Feature 2	1.21331	0	
Feature 3	1.20043	0	
Feature 6	1.19030	1	
Feature 5	0.91486	0	
Feature 4	0.55512	4	False Negative

Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Features 1 and 4)
Feature 1	**1.30836**	**9**	True Positive
Feature 2	1.05609	0	
Feature 3	1.05425	0	
Feature 6	1.03783	1	
Feature 5	0.77818	4	
Feature 4	0.46416	0	False Negative

Features are ordered based on *Velocity* rate

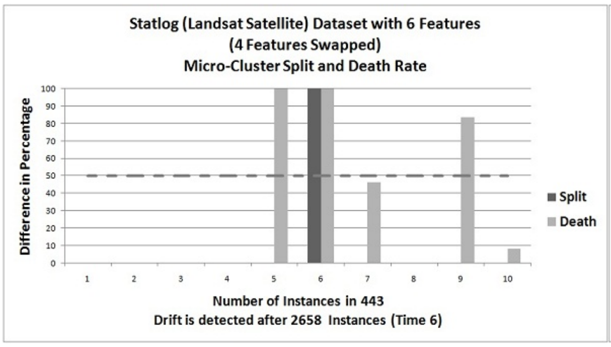
Feature	(Time 9) Velocity Rate (Descending Order)	(Time 9) History of Maximum Variance	Detection of Swapped Features (Features 1 and 4)
Feature 5	1.06363	0	
Feature 2	1.05495	0	
Feature 6	1.05404	0	
Feature 3	1.00879	0	
Feature 1	0.98005	3	False Negative
Feature 4	0.59948	0	False Negative

Features are ordered based on *Velocity* rate

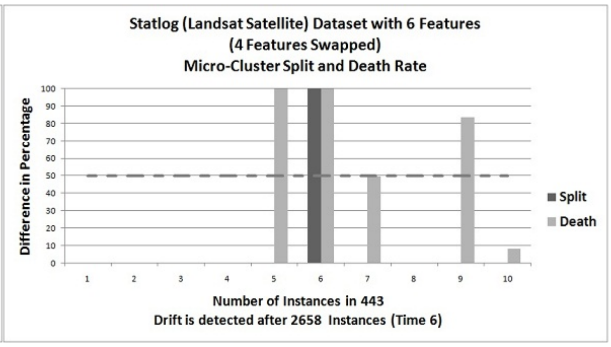
Feature	(Time 9) Velocity Rate (Descending Order)	(Time 9) History of Maximum IQR	Detection of Swapped Features (Features 1 and 4)
Feature 5	0.70381	0	
Feature 6	0.67038	5	False Positive*
Feature 2	0.66419	0	
Feature 3	0.62604	12	
Feature 1	0.59303	24	False Negative
Feature 4	0.39233	0	False Negative

Figure 5.54: The results of *Statlog (Landsat Satellite)* dataset with 6 features (2 features swapped) using Micro-Clusters for tracking features.

(a) Previous MC-NN
(Real-Time Normalisation and LPF)
(Variance)



(b) New MC-NN
(Real-Time Normalisation and LPF)
(IQR)



Features are ordered based on *Velocity* rate

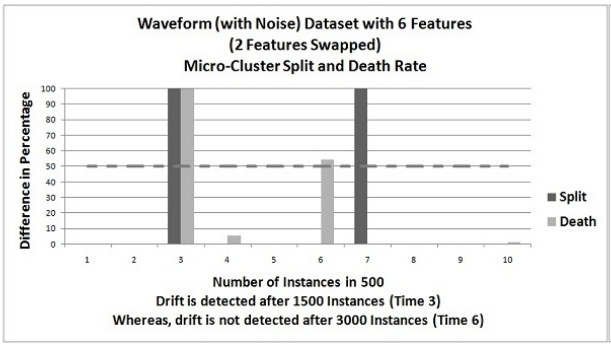
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Features 2,3,5 and 6)
Feature 5	1.03384	0	False Negative
Feature 6	0.96021	0	False Negative
Feature 3	0.72866	0	False Negative
Feature 4	0.57200	0	
Feature 1	0.55339	1	
Feature 2	0.46405	0	False Negative

Features are ordered based on *Velocity* rate

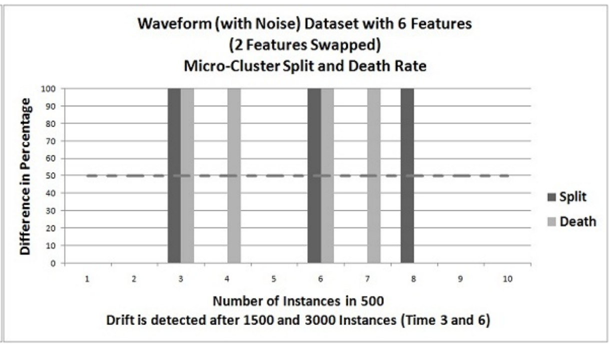
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Features 2,3,5 and 6)
Feature 5	**0.82840**	**9**	True Positive
Feature 6	0.77078	0	False Negative
Feature 3	0.58439	0	False Negative
Feature 4	0.46520	0	
Feature 1	0.38810	2	
Feature 2	0.32448	0	False Negative

Figure 5.55: The results of *Statlog (Landsat Satellite)* dataset with 6 features (4 features swapped) using Micro-Clusters for tracking features.

(a) Previous MC-NN
(Real-Time Normalisation and LPF)
(Variance)



(b) New MC-NN
(Real-Time Normalisation and LPF)
(IQR)



Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Features 1 and 4)
Feature 1	4.44609	0	False Negative
Feature 5	4.17684	0	
Feature 2	3.98962	0	
Feature 3	3.98574	0	
Feature 6	3.91226	0	
Feature 4	2.82428	0	False Negative

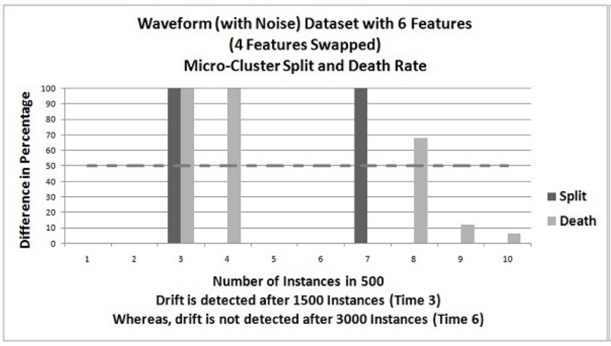
Features are ordered based on *Velocity* rate

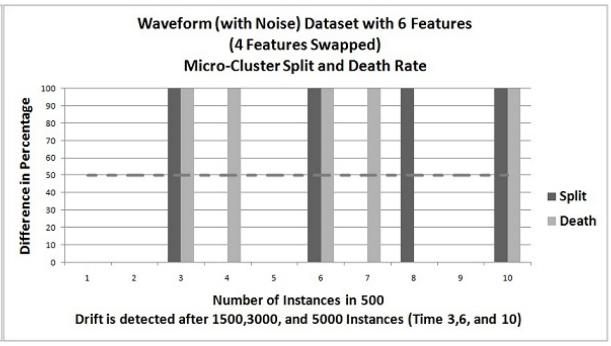
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Features 1 and 4)
Feature 4	**0.01985**	**1**	True Positive
Feature 5	0.00513	0	
Feature 1	0.00393	0	False Negative
Feature 2	0.00388	0	
Feature 3	0.00360	0	
Feature 6	0.00282	1	

Figure 5.56: The results of *Waveform (with Noise)* dataset with 6 features (2 features swapped) using Micro-Clusters for tracking features.

(a) Previous MC-NN
(Real-Time Normalisation and LPF)
(Variance)

(b) New MC-NN
(Real-Time Normalisation and LPF)
(IQR)





Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Features 2,3,5 and 6)
Feature 3	1.24006	0	False Negative
Feature 1	1.15083	0	False Negative
Feature 4	1.02329	0	
Feature 1	0.98344	0	
Feature 5	0.72471	0	False Negative
Feature 6	0.59463	0	False Negative

Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Features 2,3,5 and 6)
Feature 6	**0.02218**	**1**	True Positive
Feature 5	**0.02091**	**1**	True Positive
Feature 2	**0.00507**	**1**	True Positive
Feature 4	0.00407	0	
Feature 3	0.00373	0	False Negative
Feature 1	0.00261	0	

Figure 5.57: The results of *Waveform (with Noise)* dataset with 6 features (4 features swapped) using Micro-Clusters for tracking features.

generally achieves a much higher number of *true positive* identification of changed features compared with the method based on *Variance*.

Numbers indicated with a '*' in Table 5.3 indicate potentially *false positives* detection of features that changed their relevance. Potentially means in this case that there was also an unexpected concept drift, hence it is not possible to verify with absolute certainty if the detection was indeed a *false positive*. Whereas, numbers indicated with a (-) in Table 5.3 reflect a known drift was not detected.

Experiments in this section have been conducted on in a controlled environment on artificial datasets and in a less controlled environment on real datasets. Both methods based on *IQR* and *Variance* have been evaluated and compared on all test cases. It was observed that the method based on *IQR* identified all known concept drifts correctly. Whereas, the method based on *Variance* did miss seven known concept drifts for the controlled real data streams.

Furthermore, it was found that the method based on *IQR* identified more changed features correctly compared with the method based on *Variance*. Loosely speaking, the method based on *IQR* has shown to be superior to the method based on *Variance* in all respects. However, based on the presented results, the method based on *IQR* is chosen to evaluate the real-time feature selection method presented in the next chapter.

Table 5.3: Summary of the experimental results with controlled real datasets. The results are reported for the time of drift onset (Table 3.3) which is the time at which features were swapped.

Dataset with 6 Features	True Positive (*Variance*)	True Positive (*IQR*)	False Positive (*Variance*)	False Positive (*IQR*)
CoverType (2 features swapped)		2	2(-)	1
CoverType (4 features swapped)		2	1(-)	1
Diabetic Retinopathy Debrecen (2 features swapped)		1		1
Diabetic Retinopathy Debrecen (4 features swapped)		2	1(-)	1
Gesture Phase Segmentation (2 features swapped)		2		
Gesture Phase Segmentation (4 features swapped)		2	1(-)	
Statlog (Landsat Satellite) (2 features swapped)	1	1		1*
Statlog (Landsat Satellite) (4 features swapped)		1		
Waveform (with Noise) (2 features swapped)		1		
Waveform (with Noise) (4 features swapped)		3		
Total:	1	17	5	5

5.5 Conclusion

This chapter introduced the developed method for tracking the involved features in drifting by feeding forward the statistical information such as the *Velocity* of a feature and the spread of a feature data (i.e., *Variance* or *IQR*) provided by adaptive MC-NN's Micro-Clusters described in Chapter 3. They are calculated for each feature of a Micro-Cluster over time stamps (i.e., within a time window). Once a drift is detected which is presented in Chapter 4, the statistical information of tracked features are analysed in order to identify which features were involved in drifting for the purpose of feature selection. The analyses show that the method based on *IQR* identified more changed features correctly compared with the method based on *Variance* in all respects. In addition, tracking features would be influenced by feature-bias, outliers, and noise. Thus robustness handling or minimising the effect of feature-bias, outliers, and noise is explained in more detail in this chapter. The next chapter highlights the real-time feature selection method.

Chapter 6

Real-Time Feature Selection Method using Adaptive Micro-Clusters

The work in this chapter directly links to Objective 3 to continuously analyse the historical statistics provided by the developed method in Objective 2 with the purpose of selecting the relevant features for adaptive data stream classifier (i.e., dynamic adjustment feature selection in real-time). This chapter proposes and describes a real-time feature selection method that makes use of the drift detection method introduced in Chapter 4 and the feature tracking method developed in Chapter 5. Three main tasks will be explained in this chapter: feature analysis, feature selection, and monitoring of temporarily irrelevant features. After detection of concept drift, the statistical information of the features (i.e., *Velocity* combined with either *Variance* or *IQR*) is analysed to identify which features were involved in the drift. Loosely speaking, only features that had a significant change of their statistical information are re-examined for feature selection using Information Gain (Yang and Pedersen, 1997) in each statistical time window. This is based on the assumption that features that have not changed much are also likely to have maintained a similar Information Gain value. This can be used to reduce the computational cost of feature selection by assuming that the Information Gain has not changed for these features and thus re-calculating the Information Gain for these features is not needed. Information Gain is considered a very popular mechanism to select relevant features for a classification task which is also used in building decision trees (Han et al., 2011). Loosely speaking, the method follows the following steps which were described previously in Chapter 2 Figure 2.2 to facilitate computationally efficient feature selection:

1. Detect concept drift.
2. Use the Micro-Clusters' statistical information (*Velocity* combined with either *Variance* or *IQR*) to identify features that have been involved in the drift.
3. Re-calculate Information Gain only for features that have been involved in the drift.
4. Re-rank features according to their Information Gain.
5. Select features using the new Information Gain ranking.

The chapter is organised as follows. In Section 6.1, feature analysis and feature selection are explained. Monitoring and analysis of temporarily irrelevant features are explained in Section 6.2. A working example of applying real-time feature selection technique is explained in Section 6.3. The performance, implementation, and results of the developed method are discussed in Section 6.4 with respect to real datasets. In Section 6.5, a summary of this chapter is presented.

6.1 Feature Analysis and Feature Selection

Feature analysis is facilitated using historical data (i.e. a statistical window between *time-1* and point-of-drift *time*). For the purpose of feature analysis, a counter is kept for each feature of a Micro-Cluster over time stamps and starting with 0 within each time window. The counter is incremented by 1 if the specific feature was the feature with the highest *Variance* or *IQR*. In this research, this is referred to as the history of maximum *Variance* or *IQR*. Regarding *Velocity*, using a windowing approach over the data stream, a running average of *Velocity* rate is calculated for each feature over time windows. A high *Velocity* rate combined with the history of maximum *Variance* or *IQR* during a concept drift indicates that the feature has changed. The assumption here is that this specific feature may have changed its contribution towards the classification technique. Thus, feature selection can be limited to only examining features that have changed their *Velocity* rate as well as their *Variance* or *IQR* when there is a concept drift detected.

Features that have changed are temporarily regarded as irrelevant for the data mining task, as shown in Algorithm 5, as large statistical changes will have a similar effect as noise. However, the contributions of these features towards the absorption in Micro-Clusters may stabilise after the drift and thus are re-examined in the following time window. This will be explained in more detail in the next section. From then onwards only instances comprising the relevant features are passed on to a data stream classifier.

Algorithm 5 Feature analysis and feature selection.

Input: Micro-Clusters statistical information (*Velocity* rate [features] and history of maximum *Variance* or *IQR* [features]) of a statistical window between *time-1* and point-of-drift *time*, and instance.

Output: Instance with relevant features

```
1:  for each drift detection do
2:      List Velocity rate[features] in order from low to high
3:      Identify median value of the ordered list Velocity rate[features]
4:      for each feature with Velocity rate > median value do
5:          if feature has history of maximum Variance or IQR > 0 value then
6:              $instance \Leftarrow delete\ irrelevant\ feature$
7:          end if
8:      end for
9:  end for
10: $Classifier \Leftarrow instance\ with\ relevant\ feature(s)$
```

6.2 Monitoring and Analysis of Temporarily Irrelevant Features

The features (the ones that are temporarily regarded as irrelevant) are monitored in the following statistical window to examine their relevance using Information Gain. The assumption here is that the contribution of this specific feature towards the classification result may have changed significantly due to the drift. Thus checking relevance can be limited to examining only the features that have been temporarily regarded as irrelevant. If a feature $x's$ Information Gain is greater than μ of Information Gain between window at *time-1* and at current *time* with a *Percentage Difference* greater than 50%, then x is selected as relevant to a data stream classifier.

It should be noted here that Information Gain has been chosen as a feature selection metric, as it is a popular metric for this purpose (Han et al., 2011). Thus all experiments in the next chapter have been obtained using this metric. However, the user may decide to implement a different metric for feature selection if appropriate for the specific application. Likewise the *Percentage Difference* can be adjusted by the user requirements. However, in the experiments presented in the next chapter a *Percentage Difference* of 50% was used as it worked well in most cases as examined in Appendix C. Figure 6.1 puts the aforementioned procedure into a flowchart for a better understanding.

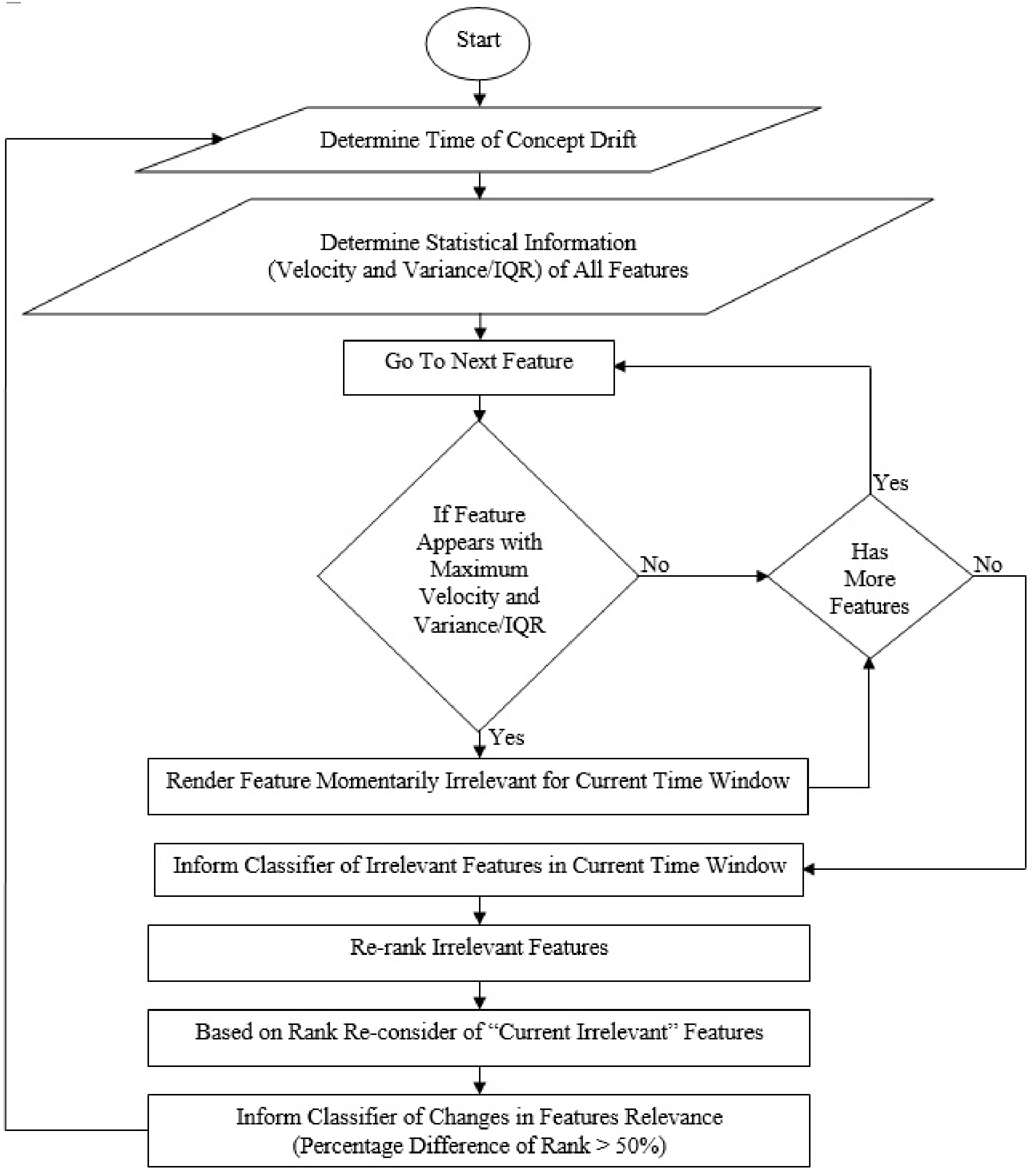
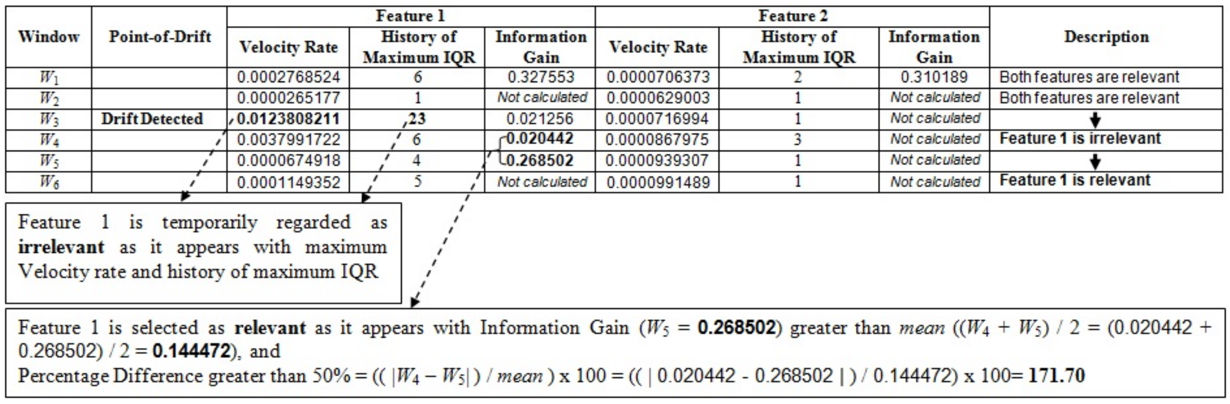

Start
Determine Time of Concept Drift
Determine Statistical Information (Velocity and Variance/IQR) of All Features
Go To Next Feature
If Feature Appears with Maximum Velocity and Variance/IQR
No
Yes
Has More Features
Yes
No
Render Feature Momentarily Irrelevant for Current Time Window
Inform Classifier of Irrelevant Features in Current Time Window
Re-rank Irrelevant Features
Based on Rank Re-consider of "Current Irrelevant" Features
Inform Classifier of Changes in Features Relevance (Percentage Difference of Rank > 50%)

Figure 6.1: Process of feature selection in real-time.

6.3 Worked Example

Figure 6.2 shows an example of the statistical information (i.e., *Velocity* rate and *IQR* as well as Information Gain) of two features for six time windows. However, *IQR* is chosen here instead of *Variance* due to the fact that *IQR* is considered robust to outliers contrary the using *Variance* as explained in Chapter 5.



Window	Point-of-Drift	Feature 1			Feature 2			Description
		Velocity Rate	History of Maximum IQR	Information Gain	Velocity Rate	History of Maximum IQR	Information Gain	
W_1		0.0002768524	6	0.327553	0.0000706373	2	0.310189	Both features are relevant
W_2		0.0000265177	1	*Not calculated*	0.0000629003	1	*Not calculated*	Both features are relevant
W_3	**Drift Detected**	**0.0123808211**	**23**	0.021256	0.0000716994	1	*Not calculated*	↓
W_4		0.0037991722	6	**0.020442**	0.0000867975	3	*Not calculated*	**Feature 1 is irrelevant**
W_5		0.0000674918	4	**0.268502**	0.0000939307	1	*Not calculated*	↓
W_6		0.0001149352	5	*Not calculated*	0.0000991489	1	*Not calculated*	**Feature 1 is relevant**

Feature 1 is temporarily regarded as **irrelevant** as it appears with maximum Velocity rate and history of maximum IQR

Feature 1 is selected as **relevant** as it appears with Information Gain (W_5 = **0.268502**) greater than *mean* ((W_4 + W_5) / 2 = (0.020442 + 0.268502) / 2 = **0.144472**), and
Percentage Difference greater than 50% = (($|W_4 - W_5|$) / *mean*) x 100 = ((| 0.020442 - 0.268502 |) / 0.144472) x 100= **171.70**

Figure 6.2: An example of feature analysis, feature selection, and monitoring of temporarily irrelevant features. Assumed Information Gains are indicated in italics, and actual Information Gain calculations are not in italics.

Initially, Information Gain is calculated for all features at window W_1. In this example, it is assumed that both features are initially relevant.

Drift is detected at window W_3, consider the *Velocity* rate and history of maximum *IQR* of Feature 1 equal to 0.0123808211 and 23, respectively. This represents that the Feature 1 appears with maximum *Velocity* rate and a history of maximum *IQR* compared with the remaining features (in this case only Feature 2). Thus Feature 1 is temporarily regarded as irrelevant to the classifier after W_3. Information gains of irrelevant features are then calculated in every time window in order to monitor if they return to being relevant. In this case, Feature 1 appears again with an Information Gain at W_5 = 0.268502, which differs from μ by 50%. This indicates that the Feature 1 becomes relevant again for classification tasks after W_5. Please note that the Information Gain of features that have remained relevant are assumed not to have changed and thus are only re-calculated if there is a drift detected. In this example assumed Information Gains are indicated in italics, and actual Information Gain calculations are not in italics. For example, for Feature 2 there is only one Information Gain calculation at the beginning in W_1. However, should there be another concept drift in the future and Feature 2 appears with maximum *Velocity* rate and history of maximum *IQR*, then its Information Gain is likely to have changed and thus it would be re-calculated.

6.4 Empirical Evaluation of Real-Time Feature Selection Method

In the previous chapters, the research study examined the capability of the developed methods to identify concept drifts and to track whether features have undergone a significant change.

The purpose was to use these methods to re-examine the relevance of the features for the classification task. If a feature $x's$ Information Gain is greater than μ of Information Gain between window at *time-1* and at current *time* with *Percentage Difference* greater than 50% it is then selected as relevant to a classifier. Where 50% yielded good results as examined in Appendix C. It has been found that the method based on *IQR* performed best in all respects. Thus the experiments in this section are conducted with the method based on *IQR*. This section evaluates the use of the real-time pre-processing technique in specific real-time feature selection method.The method is applied on a couple of real datasets with a larger number of features than in the previous more controlled experiments, with unknown ground truth, to evaluate the effect of the method in real scenarios. Information about the datasets can be found in Table 3.4 in Section 3.6.2. It should be noted that the controlled real datasets used in the previous section have been re-used for the experiments here under uncontrolled conditions. This time all features were included. It is expected that these original versions of real datasets are more challenging for data stream algorithms to induce good models compared with reduced versions used earlier in the controlled experiments in the previous section. The data stream classifier chosen to be used with the developed method was the Hoeffding Tree (Domingos and Hulten, 2000). The reason for choosing the Hoeffding Tree was due to its popularity and the fact that it is considered to be one of the most accurate data stream classifiers (Bifet et al., 2009; Bifet and Frank, 2010). The classifier (i.e., Hoeffding Tree) is training and updating incrementally instance by instance. From the start of the experiments all features are considered relevant, and the developed method may at any time during the experiment detects concept drift and flags features as relevant or irrelevant after a drift has been detected. At any time during the experiment, the classifier selects only the feature values of relevant features and accordingly uses only the currently relevant features for training and incrementally updating the model.

For the experiments in this section, two main analyses were conducted. The first analysis aims to show the accuracy differences over time the method achieved using *IQR* for concept drift detection only in comparison with applying the Hoeffding Tree classifier as a standalone method. Whereas, the second analysis aims to show the accuracy differences over time the method achieved using *IQR* for concept drift detection and real-time feature selection in comparison with applying the Hoeffding Tree classifier as a standalone method. In addition, in both analyses, real-time Min-Max *Normalisation* and *LPF* were applied to minimise the effect of feature-bias and noise, respectively.

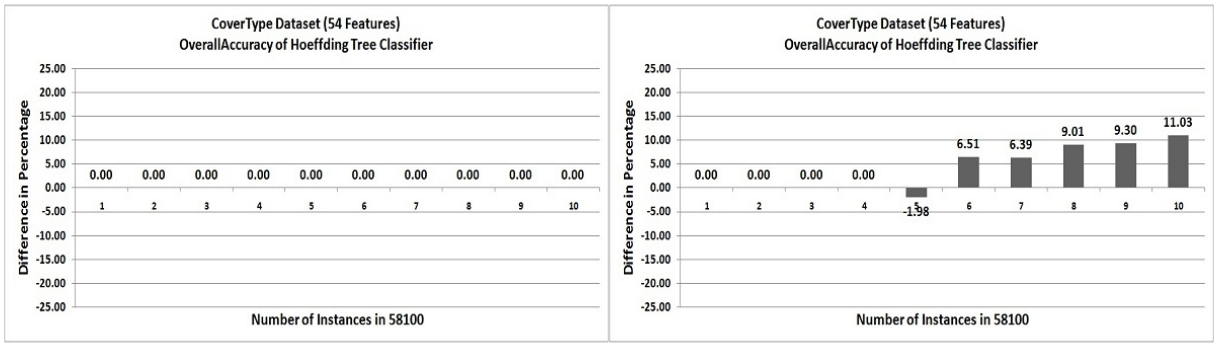
Figures (6.3 to 6.7) show the accuracy differences over time the method achieved using *IQR* for concept drift detection with/without real-time feature selection in comparison with applying the Hoeffding Tree classifier as standalone method. The figures also indicate at which time windows features and how many features have been re-evaluated for inclusion or removal from the current feature set considered by Hoeffding Tree. In order to increase the readability of the figures (i.e., results), a maximum of 25% difference is displayed. Displaying differences above 25% is not very interesting as the smallest increment/decrement in accuracy difference would indicate that the classifier has adapted positively/negatively towards the developed method. As it can be seen in the figures, Hoeffding Trees using the developed method with *IQR* for real-time feature selection generally achieved a better accuracy over time. Only during a few time windows, the method achieved a marginally lower accuracy. This potentially due to the features which were not involved in drifting but still have high *Velocity*, and they are not in the history of maximum *IQR*. A counter is kept for each feature of a Micro-Cluster over the time stamps, starting with 0 within each time window. The counter is incremented by 1 if the specific feature was the feature with the maximum *IQR* when splitting of the Micro-Cluster is performed. In this research this is referred to as the history of maximum *IQR* (i.e., see Chapter 6 Section 6.1). Hence, the features of the Micro-Cluster with high *IQR* but not the highest/maximum one are ignored, i.e., the counters of these features are not incremented by 1. Tracking these features may contribute to develop an early concept drift detection method. However, this will be examined in the future works.

Table 6.1 summarises the results for the experiments depicted in Figures (6.3 to 6.7). It states the average accuracy achieved with and without using the developed real-time feature selection method. Table 6.1 also states the average accuracy achieved by applying Hoeffding Tree classifier as a standalone method. As it can be seen a lower average accuracy was reported when Hoeffding Tree classifier is applied as a standalone method as it is unable to detect concept drift. In Hoeffding Tree, a created sub-trees can only expand from the child nodes onwards. It only keeps adapting and building trees (i.e., model) incrementally. Where a model needs to be re-initialised when concept drift is detected as described and highlighted in more detail in Chapter 4. However, in Table 6.1, the method actively re-evaluated features for inclusion in the tree at various times and achieved on average a higher accuracy compared with not employing the developed real-time feature selection method.

For the experiment, the datasets in Table 3.4 were re-conducted using NaiveBayes Classifier

(a)
(Classifier as Standalone Method)
versus
New MC-NN with
(Concept Drift Detection)

(b)
(Classifier as Standalone Method)
versus
New MC-NN with
(Concept Drift Detection and Real-Time Feature Selection)



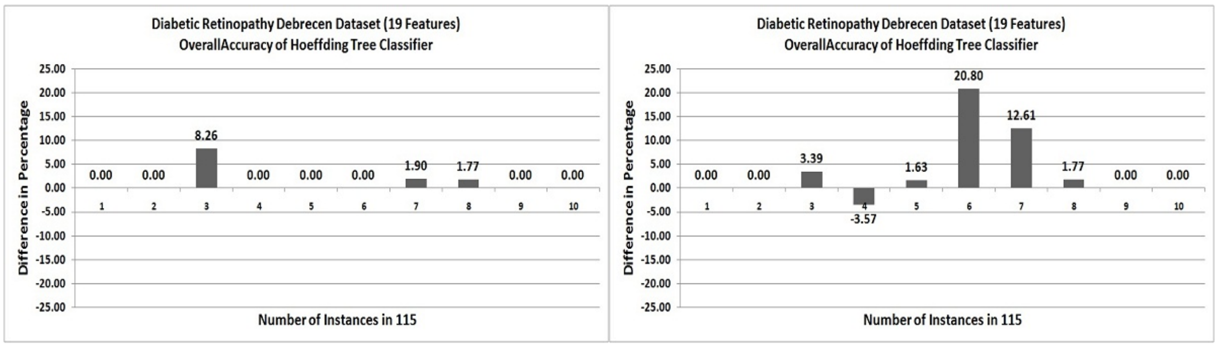
Time	Concept Drift	Index of Features Flagged	Feature Selection Task
4	Occurred	-	-

Time	Concept Drift	Index of Features Flagged	Feature Selection Task
4	Occurred	6	Velocity and Maximum IQR
7	No Drift	6	Information Gain

Figure 6.3: The results of *CoverType* dataset using Micro-Clusters for concept drift detection with/without real-time feature selection.

(a)
(Classifier as Standalone Method)
versus
New MC-NN with
(Concept Drift Detection)

(b)
(Classifier as Standalone Method)
versus
New MC-NN with
(Concept Drift Detection and Real-Time Feature Selection)



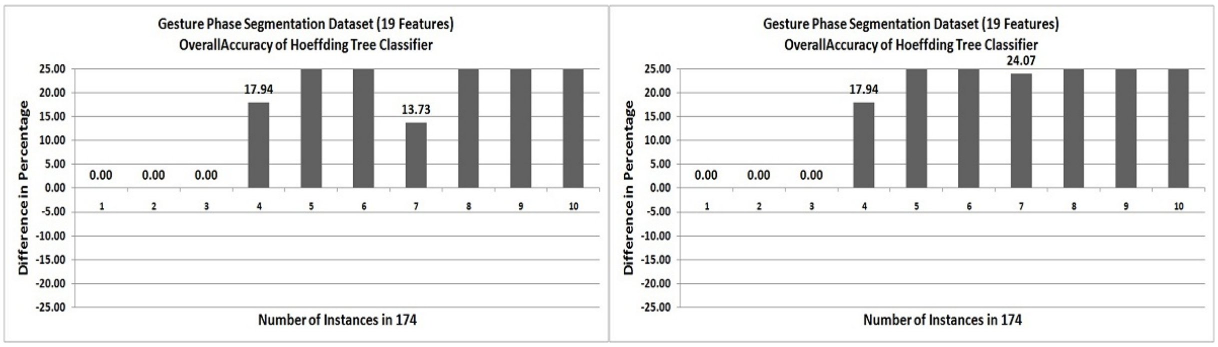
Time	Concept Drift	Index of Features Flagged	Feature Selection Task
1	Occurred	-	-

Time	Concept Drift	Index of Features Flagged	Feature Selection Task
1	Occurred	3,5,7,9,10,17, and 18	Velocity and Maximum IQR
4	No Drift	9	Information Gain
5	No Drift	3,5, and 18	Information Gain
6	No Drift	7	Information Gain
8	No Drift	10	Information Gain

Figure 6.4: The results of *Diabetic Retinopathy Debrecen* dataset using Micro-Clusters for concept drift detection with/without real-time feature selection.

(a)
(Classifier as Standalone Method)
versus
New MC-NN with
(Concept Drift Detection)

(b)
(Classifier as Standalone Method)
versus
New MC-NN with
(Concept Drift Detection and Real-Time Feature Selection)



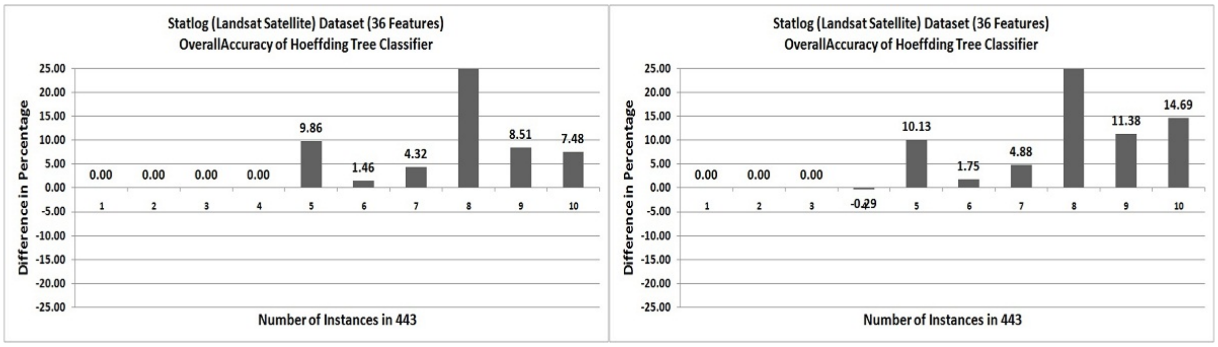
Time	Concept Drift	Index of Features Flagged	Feature Selection Task
3	Occurred	-	-

Time	Concept Drift	Index of Features Flagged	Feature Selection Task
3	Occurred	13	Velocity and Maximum IQR
8	No Drift	13	Information Gain

Figure 6.5: The results of *Gesture Phase Segmentation* dataset using Micro-Clusters for concept drift detection with/without real-time feature selection.

(a)
(Classifier as Standalone Method)
versus
New MC-NN with
(Concept Drift Detection)

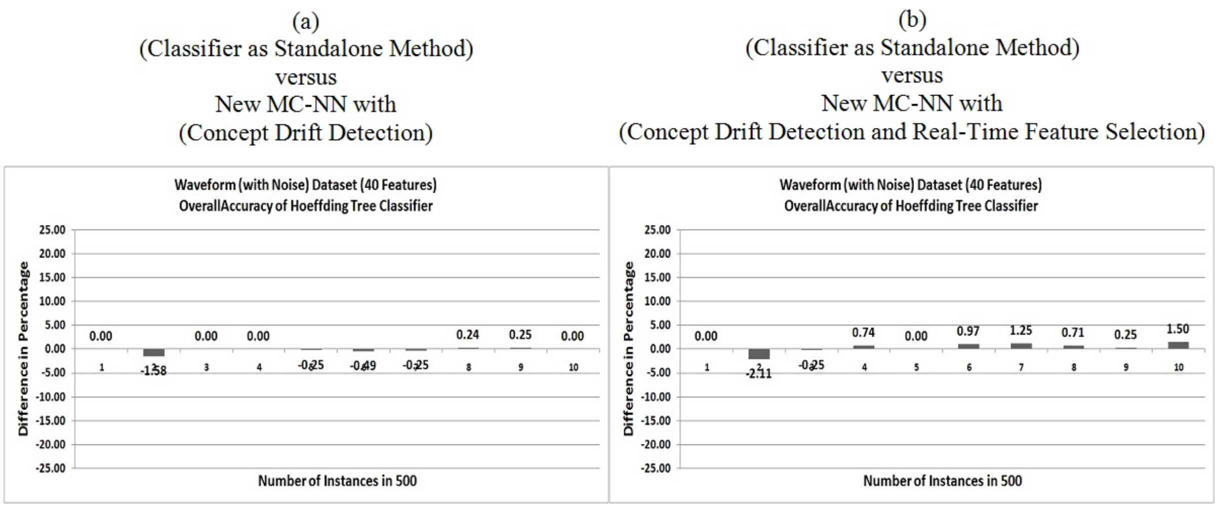
(b)
(Classifier as Standalone Method)
versus
New MC-NN with
(Concept Drift Detection and Real-Time Feature Selection)



Time	Concept Drift	Index of Features Flagged	Feature Selection Task
4	Occurred	-	-
7	Occurred	-	-

Time	Concept Drift	Index of Features Flagged	Feature Selection Task
4	Occurred	23,25, and 34	Velocity and Maximum IQR
7	Occurred	3,11, and 23	Velocity and Maximum IQR

Figure 6.6: The results of *Statlog (Landsat Satellite)* dataset using Micro-Clusters for concept drift detection with/without real-time feature selection.



Time	Concept Drift	Index of Features Flagged	Feature Selection Task
1	Occurred	-	-

Time	Concept Drift	Index of Features Flagged	Feature Selection Task
1	Occurred	15	Velocity and Maximum IQR

Figure 6.7: The results of *Waveform (with Noise)* dataset using Micro-Clusters for concept drift detection with/without real-time feature selection.

Table 6.1: Summary of the results for the experiments using real datasets with Hoeffding Tree Classifier.

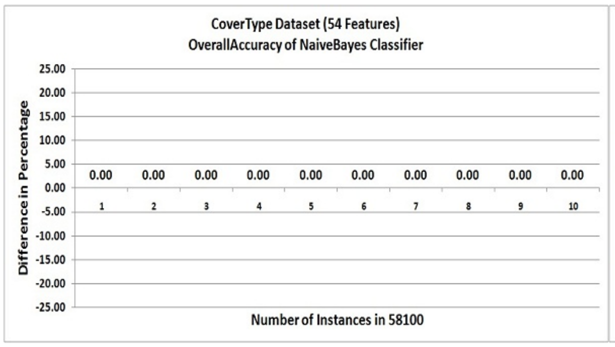
Dataset	OverallAccuracy Average with Hoeffding Tree as Standalone Method	OverallAccuracy Average with Concept Drift Detection	OverallAccuracy Average with Concept Drift Detection and Real-Time Feature Selection
CoverType	76.30	78.50	**79.32**
Diabetic Retinopathy Debrecen	49.13	49.74	**51.04**
Gesture Phase Segmentation	54.71	72.18	**80.23**
Statlog (Landsat Satellite)	76.28	80.68	**81.63**
Waveform (with Noise)	80.42	80.26	**80.68**

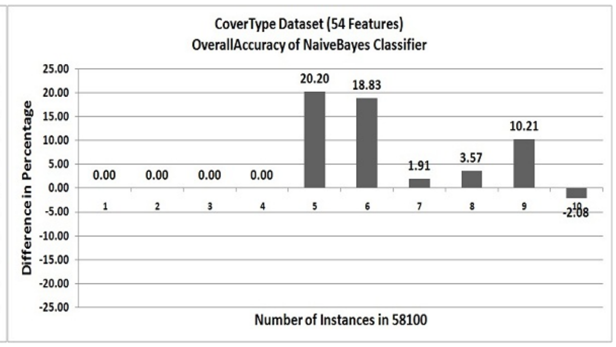
for evaluating the classification accuracy as it has high prediction rate (Sujana et al., 2017). The Bayesian rule is the main bases of the NaiveBayes classifier which assumes that the predictor features are mutually independent among the F features given the class. Figures (6.8 to 6.12) show the accuracy differences over time the method achieved using *IQR* for concept drift detection with/without real-time feature selection in comparison with applying the NaiveBayes classifier as standalone method. The figures also indicate at which time windows features and how many features have been re-evaluated for inclusion or removal from the current feature set considered by NaiveBayes. In order to increase the readability of the figures (i.e., results), a maximum of 25% difference is displayed. As it can be seen in the figures, NaiveBayes using the developed method with *IQR* for real-time feature selection generally achieved a better accuracy over time. Again only during a few time windows, the method achieved a marginally lower accuracy such as Figure 6.9, as explained before.

Table 6.2 summarises the results for the experiments depicted in Figures (6.8 to 6.12). It

(a)
(Classifier as Standalone Method)
versus
New MC-NN with
(Concept Drift Detection)

(b)
(Classifier as Standalone Method)
versus
New MC-NN with
(Concept Drift Detection and Real-Time Feature Selection)





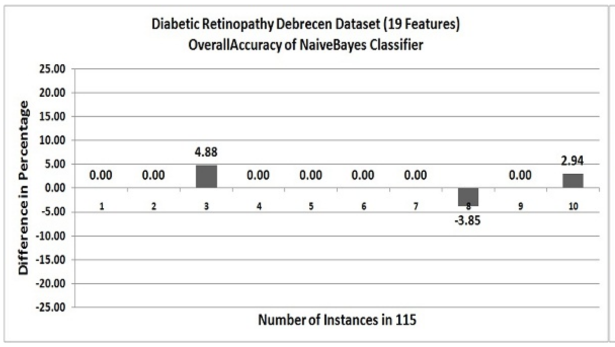
Time	Concept Drift	Index of Features Flagged	Feature Selection Task
4	Occurred	-	-

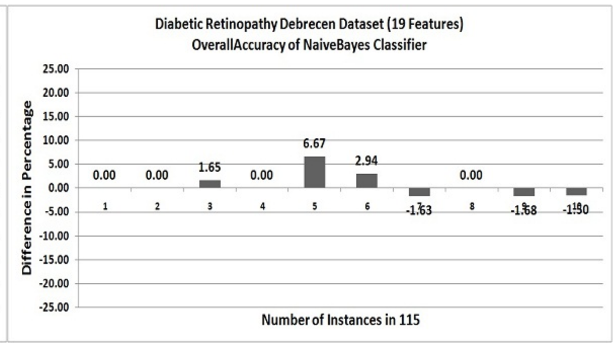
Time	Concept Drift	Index of Features Flagged	Feature Selection Task
4	Occurred	6	Velocity and Maximum IQR
7	No Drift	6	Information Gain

Figure 6.8: The results of *CoverType* dataset using Micro-Clusters for concept drift detection with/without real-time feature selection.

(a)
(Classifier as Standalone Method)
versus
New MC-NN with
(Concept Drift Detection)

(b)
(Classifier as Standalone Method)
versus
New MC-NN with
(Concept Drift Detection and Real-Time Feature Selection)





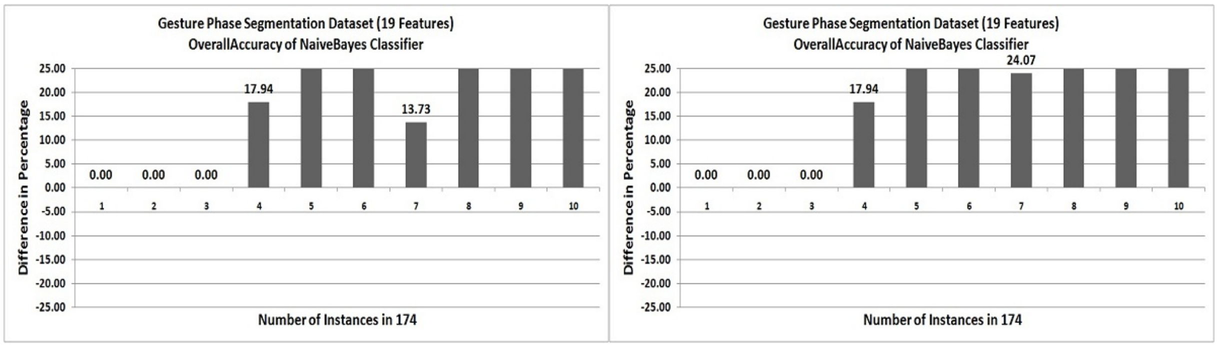
Time	Concept Drift	Index of Features Flagged	Feature Selection Task
1	Occurred	-	-

Time	Concept Drift	Index of Features Flagged	Feature Selection Task
1	Occurred	3,5,7,9,10,17, and 18	Velocity and Maximum IQR
4	No Drift	9	Information Gain
5	No Drift	3,5, and 18	Information Gain
6	No Drift	7	Information Gain
8	No Drift	10	Information Gain

Figure 6.9: The results of *Diabetic Retinopathy Debrecen* dataset using Micro-Clusters for concept drift detection with/without real-time feature selection.

(a)
(Classifier as Standalone Method)
versus
New MC-NN with
(Concept Drift Detection)

(b)
(Classifier as Standalone Method)
versus
New MC-NN with
(Concept Drift Detection and Real-Time Feature Selection)



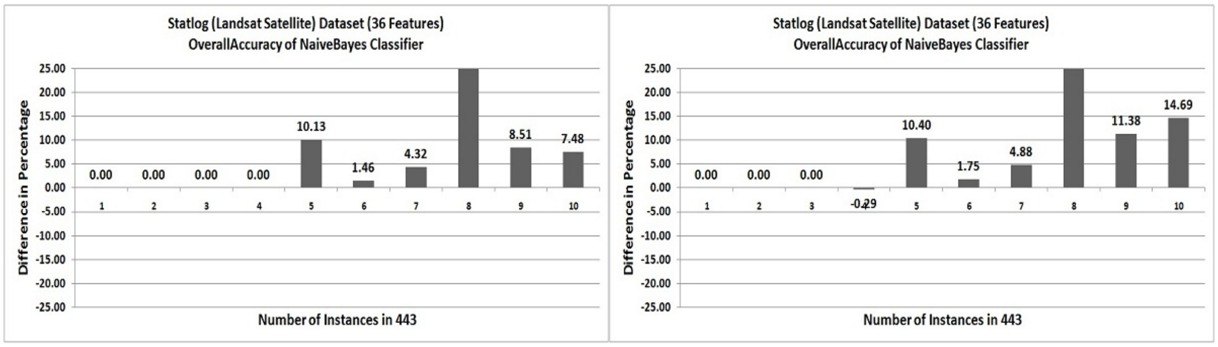
Time	Concept Drift	Index of Features Flagged	Feature Selection Task
3	Occurred	-	-

Time	Concept Drift	Index of Features Flagged	Feature Selection Task
3	Occurred	13	Velocity and Maximum IQR
8	No Drift	13	Information Gain

Figure 6.10: The results of *Gesture Phase Segmentation* dataset using Micro-Clusters for concept drift detection with/without real-time feature selection.

(a)
(Classifier as Standalone Method)
versus
New MC-NN with
(Concept Drift Detection)

(b)
(Classifier as Standalone Method)
versus
New MC-NN with
(Concept Drift Detection and Real-Time Feature Selection)



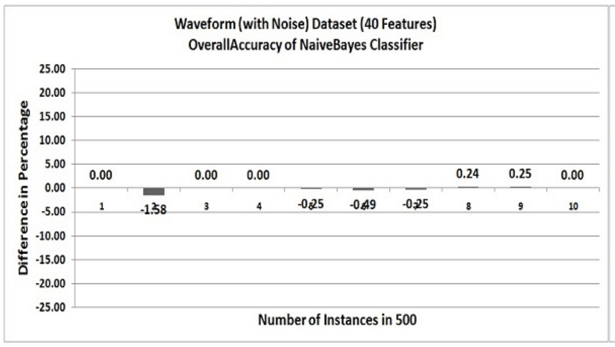
Time	Concept Drift	Index of Features Flagged	Feature Selection Task
4	Occurred	-	-
7	Occurred	-	-

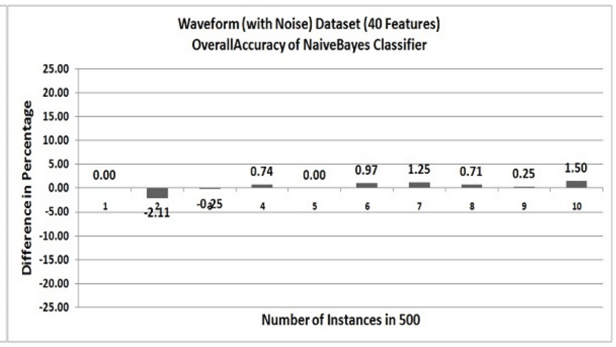
Time	Concept Drift	Index of Features Flagged	Feature Selection Task
4	Occurred	23,25, and 34	Velocity and Maximum IQR
7	Occurred	3,11, and 23	Velocity and Maximum IQR

Figure 6.11: The results of *Statlog (Landsat Satellite)* dataset using Micro-Clusters for concept drift detection with/without real-time feature selection.

(a)
(Classifier as Standalone Method)
versus
New MC-NN with
(Concept Drift Detection)

(b)
(Classifier as Standalone Method)
versus
New MC-NN with
(Concept Drift Detection and Real-Time Feature Selection)





Time	Concept Drift	Index of Features Flagged	Feature Selection Task
1	Occurred	-	-

Time	Concept Drift	Index of Features Flagged	Feature Selection Task
1	Occurred	15	Velocity and Maximum IQR

Figure 6.12: The results of *Waveform (with Noise)* dataset using Micro-Clusters for concept drift detection with/without real-time feature selection.

states the average accuracy achieved with and without using the developed real-time feature selection method. Table 6.2 also states the average accuracy achieved by applying NaiveBayes as a standalone method. As it can be seen a lower average accuracy was reported when NaiveBayes is applied as a standalone method as it is unable to detect concept drift. It only keeps adapting and building a model which needs to be re-initialised when concept drift is detected as described and highlighted in more detail in Chapter 4. However, in Table 6.2, the method actively re-evaluated features for inclusion in NaiveBayes at various times and achieved on average a higher accuracy compared with not employing the developed real-time feature selection method and only one exception with *CoverType* dataset. This potentially due to that in NaiveBayes classifier, the correlation of pairs of features given the class needs to be measured as the classification accuracy decreases when the features are not independent given the class (Friedman et al., 1997; Martinez-Arroyo and Sucar, 2006). For example, if two features have similar properties over time, i.e., Information Gain, this would indicate that these features need to be merged.

Loosely speaking, the results show that the developed real-time feature selection method indeed improves the accuracy of data stream classifiers.

Table 6.2: Summary of the results for the experiments using real datasets with NaiveBayes Classifier.

Dataset	**OverallAccuracy Average with Hoeffding Tree as Standalone Method**	**OverallAccuracy Average with Concept Drift Detection Only**	**OverallAccuracy Average with Concept Drift Detection and Real-Time Feature Selection**
CoverType	61.07	**64.31**	64.18
Diabetic Retinopathy Debrecen	51.83	52.09	**52.17**
Gesture Phase Segmentation	54.71	72.18	**80.23**
Statlog (Landsat Satellite)	76.30	80.74	**81.69**
Waveform (with Noise)	80.42	80.26	**80.68**

6.5 Conclusion

This chapter introduced the developed method for feature selection in combination with drift detection and feature tracking (Chapters 4 and 5, respectively). Adaptive Micro-Clusters were used which provide the statistical information such as a feature's speed (i.e., *Velocity*) and spread of a feature's data (i.e., *Variance* or *IQR*). The analyses show that the real-time pre-processing technique can indeed identify concept drift, track features, and identify features that may have changed their relevance for the data mining task in real-time. It has also been shown that the technique can improve the accuracy of data stream classification tasks.

Chapter 7

Conclusion and Future Works

7.1 Summary

The research investigated the problem of real-time pre-processing technique in specific real-time feature selection in combination with concept drift detection and feature tracking. At present, the focus of data stream mining lies in the development of data mining algorithms rather than on pre-processing techniques. Thus at present, there are no developments for truly real-time feature selection in a streaming setting. This is an important step in the Data Stream Mining workflow as features may potentially change their relevance for data mining tasks based on certain measures of relevance such as Information Gain. Thus the three objectives of this research were to develop a real-time pre-processing technique that can (1) detect a concept drift, (2) identify features that were involved in concept drift and thus potentially change their relevance and (3) build a real-time feature selection method based on the developments mentioned above. In this research study, the MC-NN algorithm which was developed by (Tennant et al., 2017) was used as it has been identified as a promising classification approach. MC-NN has originally been developed for predictive data stream analytics using Micro-Clusters which are able to adapt to unexpected changes in the stream. However, this research was not concerned with the classification capabilities of MC-NN but in the behaviour of its underlying model during concept drift. The MC-NN model is based on adaptive statistical Micro-Cluster summaries of the absorbed data stream instances that can split into new Micro-Clusters (Micro-Cluster *Split*), change their size/position in the feature space or be removed (Micro-Cluster *Death*) in order to adapt to concept drift. The work in this research study was based on 2 hypotheses about the behaviour of MC-NN in this regard:

(1) The *Split* and *Death* rates are expected to increase during a concept drift and thus could be used as a measure to detect concept drift in real-time; and (2) the direction of the Micro-Cluster movement in feature space during concept drift and their *Velocity* indicate which features are involved in the concept drift. This could be used as an indicator if the relevance of a feature for a data mining task has changed.

The MC-NN originally used *Variance* as a statistical measure to split the Micro-Cluster. During this research, it was expected that *Variance* as an indirect measure for concept drift adaptation would be susceptible to potential outliers and noise. Thus an alternative method has been explored for adapting Micro-Clusters based on *IQR*. Both methods have been evaluated with respect to hypothesis 1 and 2 on artificial data streams and real datasets. In addition, for both methods, Low Pass Filter (*LPF*) was also incorporated to filter out noise and *Normalisation* to reduce feature-bias. Original MC-NN did not make use of *LPF* or *Normalisation*.

Firstly, the evaluation of the developed method only focused on the method based on the already in MC-NN available statistical measure, namely *Variance*. The method was evaluated on artificial data streams and controlled real datasets with artificially induced concept drift. It was observed that the method did detect concept drifts very well on the artificial data streams compared with alternative concept drift detection methods. It achieved a very high *true positive* detection number and resulted in only five *false positive* detection. On the controlled real datasets seven artificially induced concept drifts were missed, but unknown concept drifts were detected. These unexpected concept drift detections may have been natural and thus previously unknown concept drifts but could have also been the effects of feature-bias, noise, and outliers. Thus in the next step of the evaluation *LPF*, *Normalisation*, and *IQR* as an alternative measure for feature splitting and concept drift detection were used to clean the data. To allow a fair comparison *LPF* and *Normalisation* were applied on both the method based on *Variance* and the method based on *IQR*. Both versions of the method were compared with regards to concept drift detection and feature tracking on the artificial data streams and the real datasets (with induced concept drift). On the artificial data streams, all induced concept drifts were detected correctly by the method based on *IQR* and on the controlled real datasets seven concept drifts were missed but only by the method based on *Variance*. Regarding feature tracking, the method based on *IQR* outperformed the method based on *Variance*. The method based on *IQR* achieved a high *true positive* identification of features that were actually features involved in concept drift and a low number of *false positive* identifications. Thus the method based *Variance* was considered not suitable for feature tracking, and the next step of the evaluation focussed only on the method based on *IQR* for feature selection. In this research study, the developed method for real-time feature selection was tested based on feature tracking using *IQR*. The evaluation was conducted in uncontrolled environments on five case studies with data streams based on real datasets. Thus the ground truth of feature relevance and concept drift was unknown. Hence the impact of the method on the classification accuracy over time was measured and if the method actually identified features with changing relevance correctly. The data stream classification method chosen was the popular Hoeffding Tree algorithm as well as incremental NaiveBayes. The results showed that the method detected various concept drifts throughout the streams and identified features for the re-evaluation of their relevance. It was also shown that the classifier achieved a higher average accuracy when using the developed method compared with not using the method.

Overall the research represents a first attempt to resolve real-time feature selection for data stream mining tasks. It has been shown that the method can indeed identify concept drift, track features, and identify features that may have changed their relevance for the data mining task in real-time. It has also been shown that the developed method can improve the accuracy of data stream classification tasks. This can conclude that the research work has achieved its objectives as described in Section 1.4 of this thesis according to work presented throughout the chapters of this thesis. The chapter is organised as follows. In Section 7.2, the main contributions of this research work are listed. Limitations of the study are discussed in Section 7.3. Extensions to this research study and future work in the field set out by this thesis are listed and discussed in Section 7.4.

7.2 Contributions of Research

This section presents the main contributions of this research study referring to research objectives set out in Chapter 1.

(1) A method which directly links to Objective 1 was developed and comparatively evaluated to identify a drift point (i.e., concept drift) through tracking significant changes in the statistical summaries in real-time.

(2) A method which directly links to Objective 2 was developed and evaluated to detect causality of drifts through tracking the statistics of each feature for identifying which

features were involved in drifting over a statistical time window in real-time.

(3) A method which directly links to Objective 3 was developed to continuously analyse the historical statistics provided by the developed method in Objective 2 with the purpose of selecting the relevant features for an adaptive data stream classifier (i.e., dynamic adjustment feature selection in real-time).

The above methods are embedded in a framework (i.e., the developed real-time pre-processing technique). The technique has been implemented for detecting a concept drift with the feature tracking information feedforward capability linking features to concept drifts over a statistical time window for feature selection purposes using the adaptive Micro-Clusters.

7.3 Limitations of the Study

Although this research was carefully prepared and described, shortcomings and limitations are identified and listed below.

1. The developed methods in this research were designed for tracking and analysing the relevance of features which appear with maximum statistical summaries such as *Velocity* and *IQR* when a concept drift is detected. Whereas, features with only high *Velocity* are not analysed and considered as unseen features which may have a significant effect on *OverallAccuracy* of the classifier over time.

2. The correlation of pairs of features given the class is not measured over time. This may also have a significant effect on *OverallAccuracy* of the classifier such as NaiveBayes as the classification accuracy decreases when the features are not independent given the class over time.

3. A time window size was identified in advance (i.e., fixed size, 10% of the total number of instances). Although this yielded good results in most cases, this may not be suitable for all classifiers. Also, real data streams may be potentially infinite.

7.4 Future Works

7.4.1 Extensions and Improvements

There are some extensions to this research study that would expand and strengthen the results.

1. Further enhancement to the developed methods in this area can be done by analysing the relevance of features with high *Velocity* only when concept drift is detected or occurred. It has been noticed in some results that the *OverallAccuracy* decreases over time (the evaluation of real-time feature selection, see Section 6.4). This potentially due to the features which were not involved in drifting but still have high *Velocity*, and they are not in the history of maximum *IQR*. A counter is kept for each feature of a Micro-Cluster over the time stamps, starting with 0 within each time window. The counter is incremented by 1 if the specific feature was the feature with the maximum *IQR* when splitting of the Micro-Cluster is performed. In this research this is referred to as the history of maximum *IQR* (i.e., see Chapter 6 Section 6.1). Hence, the features of the Micro-Cluster with high *IQR* but not the highest/maximum one are ignored, i.e., the counters of these features are not incremented by 1. Tracking these features may contribute to develop an early concept drift detection method.

2. Applying dynamic time window size is another possible improvement which may enhance the performance of the developed method and increase an average accuracy of the classifier such as Hoeffding Tree. Where at each node, more data instances (i.e., a greedy heuristic) would be required to optimise decisions for building tree models (Kourtellis et al., 2016).

7.4.2 Future Directions

Future work in the field set out by this thesis can be conducted in the following directions.

1. The developed technique is not embedded in a classifier as it is independent of the data stream mining algorithm (i.e., a pre-processing technique). However, a classifier can be developed by embedding the developed technique for the purpose of tracking, identifying, and detecting the best feature subset (i.e., relevant features) in real-time.

2. Another direction is to develop a real-time feature extraction method by measuring the similarity between features in terms of Information Gain as an example. If two features have similar properties, this would indicate that these features need to be merged over time to increase an average accuracy of the classifier such as NaiveBayes. Figure 7.1 shows the workflow of real-time feature extraction in predictive data stream analysis. The best candidate features from a given data stream are identified and extracted by dynamic feature extraction over time. Where Test and Train refer to Test-Then-Train or Prequential. Each individual data instance is used to test the model before it is used for training (Bifet and Frank, 2010). Examining the classification accuracy (i.e., *OverallAccuracy*) is then applied over a statistical windowing approach when *OverallAccuracy* decreases (classifier evaluation). Hence, features are examined using feature analysis which consists of three main tasks which are feature monitor, feature correlation, and feature extraction. Features are monitored using Information Gain in combination with *Velocity* and *IQR*. Correlation between features is then measured using *k*-means as an example in terms of Information Gain, *Velocity*, and *IQR*. Feature extraction merges features with similar properties. However, the merged features are monitored for the next time windows to examine their correlations over time (i.e., unmerge the features or keep them merged).

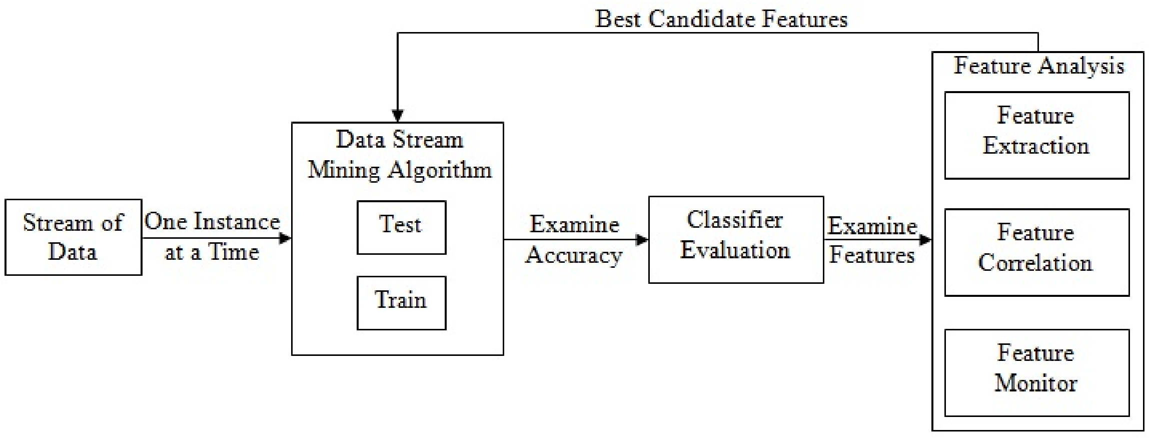


Figure 7.1: Workflow of Data Stream Mining for real-time feature extraction.

3. Outlier detection can also be achieved through tracking upper and lower limits of *IQR*. The limits of a feature's data range using *IQR* can be identified by two types *Important* and *Extreme* limits given by Equations 7.1 and 7.2 which are added to *Q*3 (i.e., upper limit) or subtracted from *Q*1 (i.e., lower limit). Figure 7.2 shows an example of *Important* and *Extreme* limits of a data's range using *IQR*. A counter is then kept for each feature of

a Micro-Cluster over time stamps and starting with 0 within time windows. The counter is incremented by 1 if the specific feature was the feature with the highest limit which is either greater or smaller than *Important* or *Extreme* limit in the current time window. A high limit (i.e., *Important* or *Extreme* limit) indicates that the feature would be affected or influenced by outliers which may have changed its contribution towards the classification task.

$$ImportantLimit[x] = 1.5 * IQR[x] \tag{7.1}$$

$$ExtremeLimit[x] = 3 * IQR[x] \tag{7.2}$$

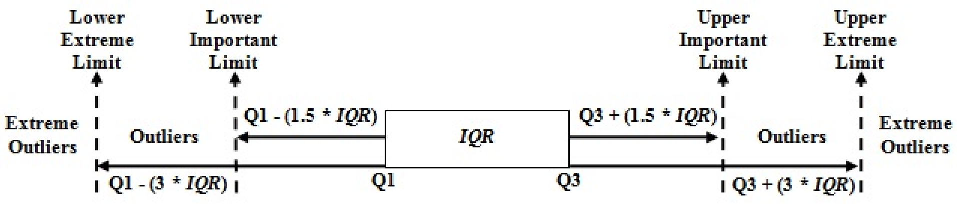


Figure 7.2: Example of important and extreme limits of feature data's range using *IQR*.

References

Aggarwal, C. and Yu, S. (2005). An effective and efficient algorithm for high-dimensional outlier detection. *The VLDB Journal—The International Journal on Very Large Data Bases*, 14(2):211–221.

Aggarwal, C. C. (2007). *Data streams: models and algorithms*, volume 31. Springer Science & Business Media.

Aggarwal, C. C. (2014). *A Survey of Stream Classification Algorithms.*

Aggarwal, C. C., Han, J., Wang, J., and Yu, P. S. (2003). A framework for clustering evolving data streams. In *Proceedings of the 29th international conference on Very large data bases-Volume 29*, pages 81–92. VLDB Endowment.

Aggarwal, C. C., Han, J., Wang, J., and Yu, P. S. (2004a). A framework for projected clustering of high dimensional data streams. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 852–863. VLDB Endowment.

Aggarwal, C. C., Han, J., Wang, J., and Yu, P. S. (2004b). On demand classification of data streams. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 503–508. ACM.

Aggarwal, C. C. and Yu, P. S. (2001). Outlier detection for high dimensional data. In *ACM Sigmod Record*, volume 30, pages 37–46. ACM.

Ahsan, U. and Essa, I. (2014). Clustering social event images using kernel canonical correlation analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 800–805.

Amini, A., Wah, T. Y., and Saboohi, H. (2014). On density-based data streams clustering algorithms: a survey. *Journal of Computer Science and Technology*, 29(1):116–141.

Amini, A., Wah, T. Y., Saybani, M. R., and Yazdi, S. R. A. S. (2011). A study of density-grid based clustering algorithms on data streams. In *Fuzzy Systems and Knowledge Discovery (FSKD), 2011 Eighth International Conference on*, volume 3, pages 1652–1656. IEEE.

Arulampalam, M. S., Maskell, S., Gordon, N., and Clapp, T. (2002). A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on signal processing*, 50(2):174–188.

Babcock, B., Babu, S., Datar, M., Motwani, R., and Widom, J. (2002). Models and issues in data stream systems. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–16. ACM.

Baena-Garcıa, M., del Campo-Ávila, J., Fidalgo, R., Bifet, A., Gavalda, R., and Morales-Bueno, R. (2006). Early drift detection method. In *Fourth international workshop on knowledge discovery from data streams*, volume 6, pages 77–86.

Barddal, J. P., Gomes, H. M., and Enembreck, F. (2014). Sfnclassifier: a scale-free social network method to handle concept drift. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing*, pages 786–791. ACM.

Barddal, J. P., Gomes, H. M., Enembreck, F., and Barthès, J.-P. (2016). Sncstream+: Extending a high quality true anytime data stream clustering algorithm. *Information Systems*, 62:60–73.

Barros, R. S., Cabral, D. R., Gonçalves Jr, P. M., and Santos, S. G. (2017). Rddm: Reactive drift detection method. *Expert Systems with Applications*, 90:344–355.

Beniwal, S. and Arora, J. (2012). Classification and feature selection techniques in data mining. *International Journal of Engineering Research & Technology (IJERT)*, 1(6).

Bhowmik, S. and Roy, C. (2007). Comparison of estimation techniques using kalman filter and grid-based filter for linear and non-linear system. In *Computing: Theory and Applications, 2007. ICCTA'07. International Conference on*, pages 516–520. IEEE.

Bifet, A. (2009). Adaptive learning and mining for data streams and frequent patterns. *ACM SIGKDD Explorations Newsletter*, 11(1):55–56.

Bifet, A. and Frank, E. (2010). Sentiment knowledge discovery in twitter streaming data. In *International conference on discovery science*, pages 1–15. Springer.

Bifet, A. and Gavalda, R. (2007). Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM international conference on data mining*, pages 443–448. SIAM.

Bifet, A., Holmes, G., Kirkby, R., and Pfahringer, B. (2010). Moa: Massive online analysis. *Journal of Machine Learning Research*, 11(May):1601–1604.

Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R., and Gavaldà, R. (2009). 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.

Bones, C. C., Romani, L. A., Sousa, E. P. M. d., et al. (2015). Clustering multivariate climate data streams using fractal dimension. In *Brazilian Symposium on Databases, XXX*. Laboratório Nacional de Computação Científica-LNCC.

Bose, R. J. C., Van Der Aalst, W. M., Žliobaitė, I., and Pechenizkiy, M. (2014). Dealing with concept drifts in process mining. *IEEE transactions on neural networks and learning systems*, 25(1):154–171.

Breunig, M. M., Kriegel, H.-P., Ng, R. T., and Sander, J. (2000). Lof: identifying density-based local outliers. In *ACM sigmod record*, volume 29, pages 93–104. ACM.

Brzeziński, D. (2010). *Mining data streams with concept drift*. PhD thesis, Dept. of Computing Science and Management, Poznan University of Technology, Poznan Google Scholar.

Brzezinski, D. and Stefanowski, J. (2014). Reacting to different types of concept drift: The accuracy updated ensemble algorithm. *IEEE Transactions on Neural Networks and Learning Systems*, 25(1):81–94.

Cao, F., Estert, M., Qian, W., and Zhou, A. (2006). Density-based clustering over an evolving data stream with noise. In *Proceedings of the 2006 SIAM international conference on data mining*, pages 328–339. SIAM.

Chaovalit, P. and Gangopadhyay, A. (2009). A method for clustering transient data streams. In *Proceedings of the 2009 ACM symposium on Applied Computing*, pages 1518–1519. ACM.

Chu, F. and Zaniolo, C. (2004). Fast and light boosting for adaptive mining of data streams. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 282–292. Springer.

Dang, X. H., Lee, V. C., Ng, W. K., and Ong, K. L. (2009). Incremental and adaptive clustering stream data over sliding window. In *International Conference on Database and Expert Systems Applications*, pages 660–674. Springer.

Davis, J. J. and Clark, A. J. (2011). Data preprocessing for anomaly based network intrusion detection: A review. *Computers & Security*, 30(6):353–375.

De Barros, R. S. M., Hidalgo, J. I. G., and de Lima Cabral, D. R. (2018). Wilcoxon rank sum test drift detector. *Neurocomputing*, 275:1954–1963.

De Gregorio, D. and Di Stefano, L. (2017). Skimap: An efficient mapping framework for robot navigation. pages 2569–2576.

Ding, S., Wu, F., Qian, J., Jia, H., and Jin, F. (2015). Research on data stream clustering algorithms. *Artificial Intelligence Review*, 43(4):593–600.

Domingos, P. and Hulten, G. (2000). Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 71–80. ACM.

Dongre, P. B. and Malik, L. G. (2014). Stream data classification and adapting to gradual concept drift. *International Journal of Advance Research in Computer Science and Management Studies*, 2(3).

Du, L., Song, Q., and Jia, X. (2014). Detecting concept drift: an information entropy based method using an adaptive sliding window. *Intelligent Data Analysis*, 18(3):337–364.

Ebbers, M., Abdel-Gayed, A., Budhi, V. B., Dolot, F., Kamat, V., Picone, R., Trevelin, J., et al. (2013). *Addressing Data Volume, Velocity, and Variety with IBM InfoSphere Streams V3. 0*. IBM Redbooks.

Fan, W. and Davidson, I. (2006). Reverse testing: an efficient framework to select amongst classifiers under sample selection bias. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 147–156. ACM.

Fong, S., Wong, R., and Vasilakos, A. V. (2016). Accelerated pso swarm search feature selection for data stream mining big data. *IEEE Transactions on Services Computing*, 9(1):33–45.

Frías-Blanco, I., del Campo-Ávila, J., Ramos-Jiménez, G., Morales-Bueno, R., Ortiz-Díaz, A., and Caballero-Mota, Y. (2015). Online and non-parametric drift detection methods based on hoeffding's bounds. *IEEE Transactions on Knowledge and Data Engineering*, 27(3):810–823.

Friedman, N., Geiger, D., and Goldszmidt, M. (1997). Bayesian network classifiers. *Machine learning*, 29(2-3):131–163.

Gaber, M. M., Zaslavsky, A., and Krishnaswamy, S. (2005). Mining data streams: a review. *ACM Sigmod Record*, 34(2):18–26.

Gama, J. (2010). *Knowledge discovery from data streams*. Chapman and Hall/CRC.

Gama, J. and Gaber, M. M. (2007). *Learning from data streams: processing techniques in sensor networks*. Springer.

Gama, J., Kosina, P., et al. (2011). Learning decision rules from data streams. In *IJCAI Proceedings-International Joint Conference on Artificial Intelligence*, volume 22, page 1255.

Gama, J., Medas, P., Castillo, G., and Rodrigues, P. (2004). Learning with drift detection. In *Advances in artificial intelligence–SBIA 2004*, pages 286–295. Springer.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., and Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)*, 46(4):44.

Gao, B. and Zhang, J. (2013). Density based distribute data stream clustering algorithm. *JSW*, 8(2):435–442.

Ghesmoune, M., Lebbah, M., and Azzag, H. (2016). State-of-the-art on clustering data streams. *Big Data Analytics*, 1(1):13.

Hahsler, M., Dunham, M. H., et al. (2010). remm: Extensible markov model for data stream clustering in r. *Journal of Statistical Software*, 35(5):1–31.

Han, J., Pei, J., and Kamber, M. (2011). *Data mining: concepts and techniques*. Elsevier.

Han, Y. (2012). *Stable feature selection: theory and algorithms*. State University of New York at Binghamton.

Hassani, M., Kranen, P., and Seidl, T. (2011). Precise anytime clustering of noisy sensor data with logarithmic complexity. In *Proceedings of the Fifth International Workshop on Knowledge Discovery from Sensor Data*, pages 52–60. ACM.

Hickey, R. J. (1996). Noise modelling and evaluating learning from examples. *Artificial Intelligence*, 82(1):157–179.

Hoens, T. R., Polikar, R., and Chawla, N. V. (2012). Learning from streaming data with concept drift and imbalance: an overview. *Progress in Artificial Intelligence*, 1(1):89–101.

Hoi, S. C., Wang, J., Zhao, P., and Jin, R. (2012). Online feature selection for mining big data. In *Proceedings of the 1st international workshop on big data, streams and heterogeneous source mining: Algorithms, systems, programming models and applications*, pages 93–100. ACM.

Houle, M. E. and Nett, M. (2015). Rank-based similarity search: Reducing the dimensional dependence. *IEEE transactions on pattern analysis and machine intelligence*, 37(1):136–150.

Hu, X. (2003). Db-hreduction: A data preprocessing algorithm for data mining applications. *Applied Mathematics Letters*, 16(6):889–895.

Hu, Y.-C., Perrig, A., and Johnson, D. B. (2003). Efficient security mechanisms for routing protocolsa. In *NDSS*.

Hulten, G., Spencer, L., and Domingos, P. (2001). Mining time-changing data streams. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 97–106. ACM.

Jadhav, A., Jadhav, P., and Kulkarni, P. (2013). A novel approach for the design of network intrusion detection system (nids). In *Sensor Network Security Technology and Privacy Communication System (SNS & PCS), 2013 International Conference on*, pages 22–27. IEEE.

Janecek, A., Gansterer, W., Demel, M., and Ecker, G. (2008). On the relationship between feature selection and classification accuracy. In *New Challenges for Feature Selection in Data Mining and Knowledge Discovery*, pages 90–105.

Kadlec, P., Gabrys, B., and Strandt, S. (2009). Data-driven soft sensors in the process industry. *Computers & Chemical Engineering*, 33(4):795–814.

Kira, K. and Rendell, L. A. (1992). The feature selection problem: Traditional methods and a new algorithm. In *AAAI*, volume 2, pages 129–134.

Knorr, E. M. and Ng, R. T. (1999). Finding intensional knowledge of distance-based outliers. In *VLDB*, volume 99, pages 211–222.

Knox, E. M. and Ng, R. T. (1998). Algorithms for mining distancebased outliers in large datasets. In *Proceedings of the International Conference on Very Large Data Bases*, pages 392–403. Citeseer.

Kohavi, R. and John, G. H. (1997). Wrappers for feature subset selection. *Artificial intelligence*, 97(1):273–324.

Kononenko, I. (1994). Estimating attributes: analysis and extensions of relief. In *European conference on machine learning*, pages 171–182. Springer.

Kotsiantis, S., Kanellopoulos, D., and Pintelas, P. (2006). Data preprocessing for supervised leaning. *International Journal of Computer Science*, 1(2):111–117.

Kourtellis, N., Morales, G. D. F., Bifet, A., and Murdopo, A. (2016). Vht: Vertical hoeffding tree. In *Big Data (Big Data), 2016 IEEE International Conference on*, pages 915–922. IEEE.

Kranen, P., Assent, I., Baldauf, C., and Seidl, T. (2009). Self-adaptive anytime stream clustering. In *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*, pages 249–258. IEEE.

Kriegel, H.-P., Kröger, P., and Zimek, A. (2009). Clustering high-dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 3(1):1.

Kuncheva, L. I. and Jain, L. C. (1999). Nearest neighbor classifier: simultaneous editing and feature selection. *Pattern recognition letters*, 20(11):1149–1156.

Lakshmi, K. P. and Reddy, C. (2015). Efficient classifier generation over stream sliding window using associative classification approach. *International Journal of Computer Applications*, 115(22).

Lavanya, D. and Rani, D. K. U. (2011). Analysis of feature selection with classification: Breast cancer datasets. *Indian Journal of Computer Science and Engineering (IJCSE)*, 2(5):756–763.

Law, Y.-N. and Zaniolo, C. (2005). An adaptive nearest neighbor classification algorithm for data streams. In *Knowledge Discovery in Databases: PKDD 2005*, pages 108–120. Springer.

Lazarevic, A. and Obradovic, Z. (2001). Effective pruning of neural network classifier ensembles. In *Neural Networks, 2001. Proceedings. IJCNN'01. International Joint Conference on*, volume 2, pages 796–801. IEEE.

Le, T., Stahl, F., Gaber, M. M., Gomes, J. B., and Di Fatta, G. (2017). On expressiveness and uncertainty awareness in rule-based classification for data streams. *Neurocomputing*, 265:127–141.

Lee, G., Singanamalli, A., Wang, H., Feldman, M. D., Master, S. R., Shih, N. N., Spangler, E., Rebbeck, T., Tomaszewski, J. E., and Madabhushi, A. (2015). Supervised multi-view canonical correlation analysis (smvcca): integrating histologic and proteomic features for predicting recurrent prostate cancer. *IEEE transactions on medical imaging*, 34(1):284–297.

Leys, C., Ley, C., Klein, O., Bernard, P., and Licata, L. (2013). Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median. *Journal of Experimental Social Psychology*, 49(4):764–766.

Li, H., Wu, X., Li, Z., and Ding, W. (2013). Group feature selection with streaming features. In *2013 IEEE 13th International Conference on Data Mining*, pages 1109–1114. IEEE.

Lichman, M. (2013). UCI machine learning repository.

Liu, A., Zhang, G., and Lu, J. (2017). Fuzzy time windowing for gradual concept drift adaptation. In *Fuzzy Systems (FUZZ-IEEE), 2017 IEEE International Conference on*, pages 1–6. IEEE.

Liu, P., Wu, N., Zhu, J., Yin, J., and Zhang, W. (2006). A unified strategy of feature selection. In *International Conference on Advanced Data Mining and Applications*, pages 457–464. Springer.

Liu, Z. et al. (2011). A method of svm with normalization in intrusion detection. *Procedia Environmental Sciences*, 11:256–262.

Lloyd, S. (1982). Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137.

Loo, H. and Marsono, M. (2015). Online data stream classification with incremental semi-supervised learning. In *Proceedings of the Second ACM IKDD Conference on Data Sciences*, pages 132–133. ACM.

Lu, Y., Sun, Y., Xu, G., and Liu, G. (2005). A grid-based clustering algorithm for high-dimensional data streams. In *International Conference on Advanced Data Mining and Applications*, pages 824–831. Springer.

Marrón, D., Read, J., Bifet, A., and Navarro, N. (2017). Data stream classification using random feature functions and novel method combinations. *Journal of Systems and Software*, 127:195–204.

Martinez-Arroyo, M. and Sucar, L. E. (2006). Learning an optimal naive bayes classifier. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 3, pages 1236–1239. IEEE.

Meesuksabai, W., Kangkachit, T., and Waiyamai, K. (2011). Hue-stream: Evolution-based clustering technique for heterogeneous data streams with uncertainty. In *Advanced Data Mining and Applications*, pages 27–40. Springer.

Mousavi, M., Bakar, A. A., and Vakilian, M. (2015). Data stream clustering algorithms: A review. *Int J Adv Soft Comput Appl*, 7(3):13.

Ogasawara, E., Martinez, L. C., De Oliveira, D., Zimbrão, G., Pappa, G. L., and Mattoso, M. (2010). Adaptive normalization: A novel data normalization approach for non-stationary time series. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Oreski, D. and Klicek, B. (2015). A novel feature selection techniques based on contrast set mining. In *14th International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases (AIKED'15)*.

Page, E. S. (1954). Continuous inspection schemes. *Biometrika*, 41(1/2):100–115.

Patro, S. and Sahu, K. K. (2015). Normalization: A preprocessing stage. *arXiv preprint arXiv:1503.06462*.

Pelayo Ramirez, L. (2011). *Developing and Evaluating Methods for Mitigating Sample Selection Bias in Machine Learning*. PhD thesis, University of Alberta.

Pesaranghader, A., Viktor, H., and Paquet, E. (2017). Mcdiarmid drift detection methods for evolving data streams. *arXiv preprint arXiv:1710.02030*.

Pesaranghader, A. and Viktor, H. L. (2016). Fast hoeffding drift detection method for evolving data streams. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 96–111. Springer.

PhridviRaj, M. and GuruRao, C. (2014). Data mining–past, present and future–a typical survey on data streams. *Procedia Technology*, 12:255–263.

Ramaswamy, S., Rastogi, R., and Shim, K. (2000). Efficient algorithms for mining outliers from large data sets. In *ACM Sigmod Record*, volume 29, pages 427–438. ACM.

Ramírez-Gallego, S., Krawczyk, B., García, S., Woźniak, M., and Herrera, F. (2017). A survey on data preprocessing for data stream mining: Current status and future directions. *Neurocomputing*, 239:39–57.

Ren, J. and Ma, R. (2009). Density-based data streams clustering over sliding windows. In *Fuzzy Systems and Knowledge Discovery, 2009. FSKD'09. Sixth International Conference on*, volume 5, pages 248–252. IEEE.

Rosenholtz, R. E. and Zakhor, A. (1991). Iterative procedures for reduction of blocking effects in transform image coding. In *Electronic Imaging'91, San Jose, CA*, pages 116–126. International Society for Optics and Photonics.

Ross, G. J., Adams, N. M., Tasoulis, D. K., and Hand, D. J. (2012). Exponentially weighted moving average charts for detecting concept drift. *Pattern Recognition Letters*, 33(2):191–198.

Salazar, A., Safont, G., Soriano, A., and Vergara, L. (2012). Automatic credit card fraud detection based on non-linear signal processing. In *Security Technology (ICCST), 2012 IEEE International Carnahan Conference on*, pages 207–212. IEEE.

Schall, O., Belyaev, A., and Seidel, H.-P. (2005). Robust filtering of noisy scattered point data. In *Proceedings Eurographics/IEEE VGTC Symposium Point-Based Graphics, 2005.*, pages 71–144. IEEE.

Sebastião, R., Gama, J., and Mendonça, T. (2017). Fading histograms in detecting distribution and concept changes. *International Journal of Data Science and Analytics*, 3(3):183–212.

Shao, J., Ahmadi, Z., and Kramer, S. (2014). Prototype-based learning on concept-drifting data streams. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 412–421. ACM.

Shavit, N. and Lotan, I. (2000). Skiplist-based concurrent priority queues. In *Parallel and Distributed Processing Symposium, 2000. IPDPS 2000. Proceedings. 14th International*, pages 263–268. IEEE.

Shukla, M., Kosta, Y., and Jayswal, M. (2017). A modified approach of optics algorithm for data streams. *Engineering, Technology & Applied Science Research*, 7(2):1478–1481.

Sidhu, P. and Bhatia, M. (2015). Empirical support for concept drifting approaches: Results based on new performance metrics. *International Journal of Intelligent Systems and Applications*, 7(6):1.

Silva, J. A., Faria, E. R., Barros, R. C., Hruschka, E. R., De Carvalho, A. C., and Gama, J. (2013). Data stream clustering: A survey. *ACM Computing Surveys (CSUR)*, 46(1):13.

Street, W. N. and Kim, Y. (2001). A streaming ensemble algorithm (sea) for large-scale classification. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 377–382. ACM.

Sujana, T. S., Rao, N. M. S., and Reddy, R. S. (2017). An efficient feature selection using parallel cuckoo search and naïve bayes classifier. In *Networks & Advances in Computational Technologies (NetACT), 2017 International Conference on*, pages 167–172. IEEE.

Sun, Y. (2007). Iterative relief for feature weighting: algorithms, theories, and applications. *IEEE transactions on pattern analysis and machine intelligence*, 29(6):1035–1051.

Sunitha, L., BalRaju, M., Sasikiran, J., and Ramana, E. V. (2014). Automatic outlier identification in data mining using iqr in real-time data. *International Journal of Advanced Research in Computer and Communication Engineering*, 3(6):7255–7257.

Tang, J., Alelyani, S., and Liu, H. (2014). Feature selection for classification: A review. *Data Classification: Algorithms and Applications*, page 37.

Tennant, M., Stahl, F., Rana, O., and Gomes, J. B. (2017). Scalable real-time classification of data streams with concept drift. *Future Generation Computer Systems*.

Thoriya, D. and Shukla, M. (2015). Study of density based clustering techniques on data streams. *International Journal of Engineering Research and Application (IJERA)*, 5(2).

Udommanetanakit, K., Rakthanmanon, T., and Waiyamai, K. (2007). E-stream: Evolution-based technique for stream clustering. In *Advanced Data Mining and Applications*, pages 605–615. Springer.

Vishwanath, R., Samartha, T., Srikantaiah, K., Venugopal, K., and Patnaik, L. M. (2013). Drsp: Dimension reduction for similarity matching and pruning of time series data streams. *arXiv preprint arXiv:1312.2669*.

Wang, D., Fong, S., Wong, R. K., Mohammed, S., Fiaidhi, J., and Wong, K. K. (2017). Robust high-dimensional bioinformatics data streams mining by odr-iovfdt. *Scientific Reports*, 7:43167.

Wang, H. and Abraham, Z. (2015). Concept drift detection for streaming data. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–9. IEEE.

Wang, H., Fan, W., Yu, P. S., and Han, J. (2003). Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 226–235. ACM.

Wang, J., Zhao, P., Hoi, S. C., and Jin, R. (2014). Online feature selection and its applications. *IEEE Transactions on Knowledge and Data Engineering*, 26(3):698–710.

Wang, S., Minku, L. L., Ghezzi, D., Caltabiano, D., Tino, P., and Yao, X. (2013). Concept drift detection for online class imbalance learning. In *Neural Networks (IJCNN), The 2013 International Joint Conference on*, pages 1–10. IEEE.

Wu, X., Yu, K., Ding, W., Wang, H., and Zhu, X. (2013). Online feature selection with streaming features. *IEEE transactions on pattern analysis and machine intelligence*, 35(5):1178–1192.

Wu, X., Yu, K., Wang, H., and Ding, W. (2010). Online streaming feature selection. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 1159–1166. Citeseer.

Wu, Y., Hoi, S. C., and Mei, T. (2014). Massive-scale online feature selection for sparse ultra-high dimensional data. *arXiv preprint arXiv:1409.7794*.

Yadav, A. and Swetapadma, A. (2014). Classification of readily biodegradable molecules using principal component analysis and artificial neural network. *ARTIFICIAL INTELLIGENCE*, 1(2).

Yan, J., Zhang, B., Liu, N., Yan, S., Cheng, Q., Fan, W., Yang, Q., Xi, W., and Chen, Z. (2006). Effective and efficient dimensionality reduction for large-scale and streaming data preprocessing. *IEEE transactions on Knowledge and Data Engineering*, 18(3):320–333.

Yang, Y. and Pedersen, J. O. (1997). A comparative study on feature selection in text categorization. In *Icml*, volume 97, pages 412–420.

Yi, W., Teng, F., and Xu, J. (2016). Noval stream data mining framework under the background of big data. *Cybernetics and Information Technologies*, 16(5):69–77.

Yu, K., Ding, W., Simovici, D. A., and Wu, X. (2012). Mining emerging patterns by streaming feature selection. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 60–68. ACM.

Zhang, T., Ramakrishnan, R., and Livny, M. (1996). Birch: an efficient data clustering method for very large databases. In *ACM Sigmod Record*, volume 25, pages 103–114. ACM.

Zhang, Y. (2008). *Noise tolerant data mining*. PhD thesis, University of Vermont.

Zhao, J., Lu, K., and He, X. (2008). Locality sensitive semi-supervised feature selection. *Neurocomputing*, 71(10):1842–1849.

Zhou, A., Cao, F., Yan, Y., Sha, C., and He, X. (2007). Distributed data stream clustering: A fast em-based approach. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pages 736–745. IEEE.

Zhu, Y. and Shasha, D. (2003). Efficient elastic burst detection in data streams. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 336–345. ACM.

Žliobaitė, I. (2010). Learning under concept drift: an overview. *arXiv preprint arXiv:1010.4784*.

Appendix A

Percentage Difference of Split and Death Rates with Low Pass Filter Rate

In this appendix, different values of *Percentage Difference* of *Split* and *Death* rates in combination with α rate of *LPF* were used for the purpose of concept drift detection and real-time feature selection using artificial and real datasets (controlled and uncontrolled real datasets).

A.1 Artificial and Controlled Real Datasets

Table A.1 illustrates number of concept drifts detected correctly. For the experiments the default parameters stated in Table 3.2 (artificial datasets) and Table 3.3 (controlled real datasets with 6 features) were used unless stated otherwise. The new MC-NN with *IQR* was used. As there are 25 experiments, there is a total of 25 concept drifts to be detected. Where 15 concepts drift with artificial datasets, and 10 concepts drift with controlled datasets. It can be seen that 50% of α rate yielded good results in most cases. Whereas, *Percentage Difference* rates from 40% to 60% yielded similar results in terms of *true positive* detections (25). However, a lower rate of *Percentage Difference* (i.e., less than 50%) potentially would render the developed method triggers frequent and unnecessary adaptation to concept drift. Whereas, a higher rate of *Percentage Difference* (i.e., greater than 50%) potentially would render the developed method unable to detect actual concept drifts.

Table A.1: Summary of concept drifts adaptation experiments using artificial and controlled real dataset with different values (from 10% to 100%) of *Percentage Difference* of *Split* and *Death* rates in combination with α. The results are reported for the Time 6 which is the time at which features were swapped. Where a first number refers to number of *false positive* detections, whereas a second number (between round brackets) refers to number of *true positive* detections.

α / **Percentage Difference Rate**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**100%**
10%	41(15)	42(14)	31(16)	27(16)	17(24)	25(18)	24(18)	28(19)	26(16)	25(17)
20%	41(15)	40(12)	29(15)	26(16)	4(24)	21(19)	27(18)	26(17)	25(16)	25(17)
30%	36(14)	38(12)	27(15)	23(17)	13(24)	27(19)	23(17)	26(17)	21(16)	22(16)
40%	28(14)	31(12)	26(14)	22(16)	**10(25)**	27(19)	22(17)	24(17)	18(16)	18(16)
50%	26(14)	30(12)	21(14)	21(16)	**9(25)**	16(19)	22(16)	21(17)	18(16)	18(16)
60%	26(14)	28(12)	20(14)	20(16)	**9(25)**	17(18)	22(16)	20(17)	18(16)	18(16)
70%	25(14)	27(12)	19(14)	20(16)	9(24)	17(18)	22(16)	20(17)	17(16)	18(16)
80%	24(14)	26(12)	19(15)	18(16)	9(24)	15(17)	22(16)	20(17)	17(16)	18(16)
90%	23(14)	24(12)	18(15)	18(16)	7(24)	15(17)	18(16)	20(17)	17(16)	18(16)
100%	23(14)	23(12)	18(15)	16(16)	7(23)	14(17)	17(16)	21(16)	17(16)	16(16)

A.2 Uncontrolled Real Datasets

Tables A.2 to A.6 illustrate average accuracy achieved using uncontrolled real datasets with the developed methods for real-time feature selection and concept drift detection. For the experiments the default parameters stated in Table 3.4 (uncontrolled real datasets) were used unless stated otherwise. The new MC-NN with *IQR* was used. Hoeffding Tree classifier was used as well. Although higher average accuracy achieved in most cases, α with higher/lower rate (i.e., greater/smaller than 50%) potentially would lead to losing information. According to *LPF* equation (see Chapter 2 Section 2.3.2), α with 50% means that the method uses 50% of both a new incoming data and a feature's filtered data. Regarding *Percentage Difference* of *Split* and *Death* rates, lower rate of *Percentage Difference* (i.e., less than 50%) potentially would render the developed method triggers frequent and unnecessary adaptation to concept drifts, whereas higher rate of *Percentage Difference* (i.e., greater than 50%) potentially would render the developed method unable to detect actual concept drifts. Hence, in this research study, α and *Percentage Difference* of *Split* and *Death* rates are set to 50%.

Table A.2: Summary of average accuracy achieved using uncontrolled real dataset (*CoverType*) with different values (from 10% to 100%) of *Percentage Difference* of *Split* and *Death* rates in combination with α.

α **Percentage Difference Rate**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**100%**
10%	78.29	79.85	78.86	**80.00**	79.91	78.91	78.91	79.48	78.70	78,58
20%	78.29	79.85	78.86	**80.00**	79.32	78.91	78.91	79.48	78.70	78,58
30%	78.29	79.73	79.73	79.68	79.32	79.11	78.91	79.48	78.70	78,58
40%	76.30	79.73	79.73	79.68	79.32	79.11	78.91	79.48	78.70	78,58
50%	76.30	79.73	79.73	79.68	79.32	79.11	78.91	79.48	78.70	78,58
60%	76.30	79.73	79.73	76.30	79.32	79.11	78.91	79.48	78.70	78,58
70%	76.30	76.30	76.30	76.30	79.32	79.11	78.91	79.48	78.70	78,58
80%	76.30	76.30	76.30	76.30	79.32	79.11	78.91	79.48	78.70	78,58
90%	76.30	76.30	76.30	76.30	79.32	79.11	78.91	79.48	78.70	78,58
100%	76.30	76.30	76.30	76.30	79.32	79.11	78.91	79.48	78.97	78,58

Table A.3: Summary of average accuracy achieved using uncontrolled real dataset (*Diabetic Retinopathy Debrecen*) with different values (from 10% to 100%) of *Percentage Difference* of *Split* and *Death* rates in combination with α.

α **Percentage Difference Rate**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**100%**
10%	50.09	50.35	50.00	50.70	50.43	**52.52**	51.22	**52.52**	51.39	49.83
20%	50.09	50.35	50.00	50.70	51.48	**52.52**	51.22	**52.52**	51.39	49.83
30%	50.09	50.35	50.00	52.17	51.48	**52.52**	51.22	**52.52**	51.22	49.83
40%	50.09	50.35	50.00	52.17	51.48	**52.52**	51.22	**52.52**	50.87	49.83
50%	50.09	50.35	50.00	52.17	51.04	52.00	51.22	52.43	50.87	49.83
60%	50.09	50.35	50.00	52.17	51.04	52.00	51.22	52.43	50.87	49.83
70%	50.09	50.35	50.00	52.17	51.04	52.09	51.22	52.43	50.87	49.83
80%	50.09	50.35	50.00	52.17	51.04	52.09	51.22	52.43	50.87	49.83
90%	50.09	50.35	50.00	52.17	51.04	52.09	51.22	52.43	50.87	49.83
100%	50.09	50.35	50.00	52.17	51.04	52.09	51.22	52.43	50.87	49.83

Table A.4: Summary of average accuracy achieved using uncontrolled real dataset (*Gesture Phase Segmentation*) with different values (from 10% to 100%) of *Percentage Difference* of *Split* and *Death* rates in combination with α.

α **Percentage Difference Rate**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**100%**
10%	74.46	74.48	81.03	75.80	**81.38**	81.21	83.45	75.28	79.31	79.48
20%	77.41	77.70	81.03	75.80	**81.38**	81.21	75.80	74.89	79.31	79.48
30%	76.67	72.99	73.10	75.80	76.32	81.21	75.80	74.89	79.31	79.48
40%	76.67	72.99	73.10	75.80	80.00	81.21	75.80	74.89	79.31	79.48
50%	77.64	72.99	75.86	72.01	80.23	80.46	75.80	74.89	79.31	79.48
60%	77.64	72.99	75.86	70.52	54.71	80.46	75.80	74.89	79.31	79.48
70%	77.64	72.99	75.86	70.52	54.71	61.03	75.80	74.89	79.31	79.48
80%	77.70	72.99	75.86	70.52	54.71	61.03	75.80	74.89	79.31	79.48
90%	77.70	72.99	75.86	70.52	54.71	61.03	75.80	74.89	79.31	79.48
100%	77.70	72.99	75.86	70.52	54.71	61.03	75.80	74.89	79.31	79.08

Table A.5: Summary of average accuracy achieved using uncontrolled real dataset (*Statlog (Landsat Satellite)*) with different values (from 10% to 100%) of *Percentage Difference* of *Split* and *Death* rates in combination with α.

α / **Percentage Difference Rate**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**100%**
10%	**83.16**	82.00	82.10	80.79	80.74	82.23	81.49	81.99	80.41	81.47
20%	**83.16**	82.00	82.10	80.79	80.74	82.21	80.59	81.99	80.41	81.47
30%	**83.16**	82.00	82.10	80.79	80.74	82.21	80.59	81.99	80.41	80.41
40%	79.12	82.53	78.02	80.79	81.63	82.21	80.59	81.99	80.41	80.41
50%	78.53	77.86	78.08	79.59	81.63	79.37	80.59	82.14	80.41	80.41
60%	78.53	77.86	78.08	79.59	79.50	79.37	80.50	82.14	80.41	80.41
70%	78.53	77.86	78.08	79.59	79.50	79.37	80.50	82.14	80.41	80.41
80%	78.53	77.86	78.08	79.59	79.50	79.37	80.45	82.14	80.41	80.41
90%	78.53	77.86	78.08	79.59	79.50	79.37	80.45	82.14	80.41	80.41
100%	78.53	77.86	78.08	79.59	79.50	79.37	80.45	82.14	80.41	80.41

Table A.6: Summary of average accuracy achieved using uncontrolled real dataset (*Waveform (with noise)*) with different values (from 10% to 100%) of *Percentage Difference* of *Split* and *Death* rates in combination with α.

α / **Percentage Difference Rate**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**100%**
10%	77.70	72.58	76.24	78.88	78.84	80.68	80.42	80.42	80.42	80.42
20%	79.40	77.14	79.40	79.46	79.36	80.42	80.42	80.42	80.42	80.42
30%	**80.96**	80.69	80.26	80.26	80.68	80.42	80.42	80.42	80.42	80.42
40%	**80.96**	**80.96**	80.26	80.26	80.68	80.42	80.42	80.42	80.42	80.42
50%	**80.96**	**80.96**	80.26	80.26	80.68	80.42	80.42	80.42	80.42	80.42
60%	**80.96**	**80.96**	80.26	80.26	80.68	80.42	80.42	80.42	80.42	80.42
70%	**80.96**	**80.96**	80.26	80.26	80.68	80.42	80.42	80.42	80.42	80.42
80%	**80.96**	**80.96**	80.26	80.26	80.68	80.42	80.42	80.42	80.42	80.42
90%	**80.96**	**80.96**	80.26	80.26	80.68	80.42	80.42	80.42	80.42	80.42
100%	**80.96**	**80.96**	80.26	80.26	80.68	80.42	80.42	80.42	80.42	80.42

Appendix B

FIFO Queue Size

In this appendix, different *FIFO*'s sizes (i.e., number of instances saved in the *FIFO*) were used for splitting a Micro-Cluster using *IQR* with artificial and real datasets (controlled and uncontrolled real datasets). For the experiments in this appendix, α rate of *LPF* and *Percentage Difference* of *Split* and *Death* rates were also used with *FIFO* queue. Regarding α and *Percentage Difference* rates, the same results as the previous appendix (Appendix A) have been achieved. 50% of α and *Percentage Difference* rates yielded good results in most cases. Whereas, *FIFO*'s sizes (500 and 1000) yielded good results in most cases. However, lower *FIFO*'s size (i.e., 100) potentially would lead to losing information. Whereas, higher *FIFO*'s size (i.e., greater than 1000) potentially may consider computationally expensive.

B.1 FIFO Queue in combination with Percentage Difference of Split and Death Rates

B.1.1 Artificial and Controlled Real Datasets

Table B.1 illustrates number of concept drifts detected correctly using different *FIFO*'s sizes have been selected randomly (100, 500, and 1000) in combination with different values of *Percentage Difference* of *Split* and *Death* rates (from 10% to 100%). For the experiments the default parameters stated in Table 3.2 (artificial datasets) and Table 3.3 (controlled real datasets) were used. The new MC-NN with *IQR* was used. α rate of *LPF* was set to 50%. As there are 25 experiments, there is a total of 25 concept drifts to be detected. Where 15 concepts drifts with artificial datasets, and 10 concepts drifts with controlled datasets.

Table B.1: Summary of concept drifts adaptation experiments using artificial and controlled real dataset with different values (from 10% to 100%) of *Percentage Difference* of *Split* and *Death* rates in combination with different *FIFO*'s sizes (100, 500, and 1000). The results are reported for the Time 6 which is the time at which features were swapped. Where a first number refers to number of *false positive* detections, whereas a second number (between round brackets) refers to number of *true positive* detections.

Percentage Difference Rate / ***FIFO*'s Size**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**100%**
100	27(19)	24(20)	24(19)	23(20)	21(20)	23(19)	21(19)	21(19)	20(19)	18(18)
500	20(24)	16(24)	16(23)	16(23)	14(24)	14(24)	14(24)	14(24)	13(24)	13(23)
1000	23(24)	18(24)	18(23)	**15(25)**	**14(25)**	**14(25)**	14(24)	14(24)	13(24)	12(23)

B.1.2 Uncontrolled Real Datasets

Tables B.2 to B.6 illustrate average accuracy achieved using uncontrolled real datasets with the developed methods for real-time feature selection and concept drift detection. For the experiments the default parameters stated in Table 3.4 (uncontrolled real datasets) were used unless stated otherwise. The new MC-NN with *IQR* and Hoeffding Tree classifier were used.

Table B.2: Summary of average accuracy achieved using uncontrolled real dataset (*CoverType*) with different values (from 10% to 100%) of *Percentage Difference* of *Split* and *Death* rates in combination with different *FIFO*'s sizes (100, 500, and 1000).

Percentage Difference Rate / ***FIFO*'s Size**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**100%**
100	79.14	79.14	79.35	79.35	79.35	79.35	79.35	79.35	79.35	79.35
500	79.24	79.24	79.32	79.32	79.32	79.32	79.32	79.32	79.32	79.32
1000	**79.91**	79.32	79.32	79.32	79.32	79.32	79.32	79.32	79.32	79.32

Table B.3: Summary of average accuracy achieved using uncontrolled real dataset (*Diabetic Retinopathy Debrecen*) with different values (from 10% to 100%) of *Percentage Difference* of *Split* and *Death* rates in combination with different *FIFO*'s sizes (100, 500, and 1000).

Percentage Difference Rate / ***FIFO*'s Size**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**100%**
100	49.65	51.13	51.13	51.04	50.70	50.70	50.70	50.70	50.70	50.70
500	50.43	**51.48**	**51.48**	**51.48**	51.04	51.04	51.04	51.04	51.04	51.04
1000	50.43	**51.48**	**51.48**	**51.48**	51.04	51.04	51.04	51.04	51.04	51.04

Table B.4: Summary of average accuracy achieved using uncontrolled real dataset (*Gesture Phase Segmentation*) with different values (from 10% to 100%) of *Percentage Difference* of *Split* and *Death* rates in combination with different *FIFO*'s sizes (100, 500, and 1000).

Percentage Difference Rate / ***FIFO*'s Size**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**100%**
100	**81.38**	**81.38**	76.44	80.00	80.23	54.71	54.71	54.71	54.71	54.71
500	**81.38**	**81.38**	76.32	80.00	80.23	54.71	54.71	54.71	54.71	54.71
1000	**81.38**	**81.38**	76.32	80.00	80.23	54.71	54.71	54.71	54.71	54.71

Table B.5: Summary of average accuracy achieved using uncontrolled real dataset (*Statlog (Landsat Satellite)*) with different values (from 10% to 100%) of *Percentage Difference* of *Split* and *Death* rates in combination with different *FIFO*'s sizes (100, 500, and 1000).

Percentage Difference Rate / ***FIFO*'s Size**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**100%**
100	81.47	81.47	81.47	81.38	81.38	80.43	80.43	80.43	80.43	80.43
500	80.75	80.75	80.75	**81.63**	**81.63**	79.50	79.50	79.50	79.50	79.50
1000	80.75	80.75	80.75	**81.63**	**81.63**	79.50	79.50	79.50	79.50	79.50

Table B.6: Summary of average accuracy achieved using uncontrolled real dataset (*Waveform (with noise)*) with different values (from 10% to 100%) of *Percentage Difference* of *Split* and *Death* rates in combination with different *FIFO*'s sizes (100, 500, and 1000).

Percentage Difference Rate / ***FIFO*'s Size**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**100%**
100	78.54	78.74	79.92	79.92	79.92	79.92	79.92	79.92	79.92	79.92
500	78.84	79.36	**80.68**	**80.68**	**80.68**	**80.68**	**80.68**	**80.68**	**80.68**	**80.68**
1000	78.84	79.36	**80.68**	**80.68**	**80.68**	**80.68**	**80.68**	**80.68**	**80.68**	**80.68**

B.2 FIFO Queue in combination with Alpha Rate of LPF

B.2.1 Artificial and Controlled Real Datasets

Table B.7 illustrates number of concept drifts detected correctly using different *FIFO*'s sizes have been selected randomly (100, 500, and 1000) in combination with different values of α rate of *LPF* (from 10% to 100%). For the experiments the default parameters stated in Table 3.2 (artificial datasets) and Table 3.3 (controlled real datasets) were used. The new MC-NN with *IQR* was used. *Percentage Difference* of *Split* and *Death* rates was set to 50%. As there are 25 experiments, there is a total of 25 concept drifts to be detected. Where 15 concepts drifts with artificial datasets, and 10 concepts drifts with controlled datasets. *Percentage Difference* of *Split* and *Death* rates was set to 50%.

Table B.7: Summary of concept drift adaptation experiments using artificial and controlled real dataset with different values (from 10% to 100%) of α rate in combination with different *FIFO*'s sizes (100, 500, and 1000). The results are reported for the Time 6 which is the time at which features were swapped. Where a first number refers to number of *false positive* detections, whereas a second number (between round brackets) refers to number of *true positive* detections.

α / ***FIFO*'s Size**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**100%**
100	28(13)	32(12)	25(13)	18(15)	21(19)	18(17)	24(16)	26(16)	23(16)	23(15)
500	26(13)	29(14)	22(13)	21(18)	13(24)	19(22)	27(18)	20(18)	20(17)	20(17)
1000	26(14)	29(13)	21(14)	22(17)	**14(25)**	19(21)	25(18)	22(18)	19(17)	20(17)

B.2.2 Uncontrolled Real Datasets

Tables B.8 to B.12 illustrates average accuracy achieved using uncontrolled real datasets with the developed methods for real-time feature selection and concept drift detection. For the experiments the default parameters stated in Table 3.4 (uncontrolled real datasets) were used unless stated otherwise. The new MC-NN with *IQR* was used as well as Hoeffding Tree Classifier.

Table B.8: Summary of average accuracy achieved using uncontrolled real dataset (*CoverType*) with different values (from 10% to 100%) of α rate of *LPF* in combination with different *FIFO*'s sizes (100, 500, and 1000).

α / ***FIFO*'s Size**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**100%**
100	76.30	78.90	78.90	79.03	79.35	78.75	78.19	78.81	78.58	78.61
500	76.30	79.19	79.86	**80.00**	79.32	79.20	78.96	79.20	78.27	78.97
1000	76.30	79.73	79.73	**80.00**	79.32	79.12	78.91	79.48	78.70	78.58

Table B.9: Summary of average accuracy achieved using uncontrolled real dataset (*Diabetic Retinopathy Debrecen*) with different values (from 10% to 100%) of α rate of *LPF* in combination with different *FIFO*'s sizes (100, 500, and 1000).

α / ***FIFO*'s Size**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**100%**
100	50.08	51.30	50.26	51.74	50.70	51.65	51.57	52.09	50.26	49.39
500	50.08	50.35	50.00	52.17	51.04	52.00	51.22	**52.43**	50.87	49.83
1000	50.08	50.35	50.00	52.17	51.04	52.00	51.22	**52.43**	50.87	49.83

Table B.10: Summary of average accuracy achieved using uncontrolled real dataset (*Gesture Phase Segmentation*) with different values (from 10% to 100%) of α rate of *LPF* in combination with different *FIFO*'s sizes (100, 500, and 1000).

α / ***FIFO*'s Size**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**100%**
100	71.38	72.53	71.15	72.01	80.23	**80.46**	75.80	74.60	79.25	79.14
500	77.64	72.99	75.86	72.01	80.23	**80.46**	75.80	74.89	79.31	79.48
1000	77.64	72.99	75.86	72.01	80.23	**80.46**	75.80	74.89	79.31	79.48

Table B.11: Summary of average accuracy achieved using uncontrolled real dataset (*Statlog (Landsat Satellite)*) with different values (from 10% to 100%) of α rate of *LPF* in combination with different *FIFO*'s sizes (100, 500, and 1000).

α / ***FIFO*'s Size**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**100%**
100	82.05	80.59	80.77	78.08	81.38	77.79	80.51	**82.51**	81.29	80.14
500	78.53	77.86	78.08	79.59	81.63	79.37	80.59	82.14	80.41	80.41
1000	78.53	77.86	78.08	79.59	81.63	79.37	80.59	82.14	80.41	80.41

Table B.12: Summary of average accuracy achieved using uncontrolled real dataset (*Waveform (with noise)*) with different values (from 10% to 100%) of α rate of *LPF* in combination with different *FIFO*'s sizes (100, 500, and 1000).

α / ***FIFO*'s Size**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**100%**
100	80.26	80.26	80.26	80.26	79.92	80.42	80.42	80.42	80.42	80.42
500	**80.96**	**80.96**	80.26	80.26	80.68	80.42	80.42	80.42	80.42	80.42
1000	**80.96**	**80.96**	80.26	80.26	80.68	80.42	80.42	80.42	80.42	80.42

Appendix C

Percentage Difference of Information Gain for Real-Time Feature Selection

In this appendix, different values of *Percentage Difference* of Information Gain were used for the purpose of real-time feature selection in particular monitoring of temporarily irrelevant features. When a concept drift is detected, and the real-time feature selection method is applied, Information Gain is calculated based on the features' values saved in *FIFO* queues. Therefore, in this appendix, different *FIFO*'s sizes have been selected randomly (100, 500, and 1000) were used in combination with different values (from 10% to 100%) of *Percentage Difference* of Information Gain. Uncontrolled real datasets were used. The default parameters stated in Table 3.4 were used. Hoeffding Tree classifier and the new MC-NN with *IQR* were used. For the experiments in this appendix, the α rate of *LPF* and *Percentage Difference* of *Split* and *Death* rates were set to 50% as it yielded good results in most cases as examined in the previous appendices. Tables C.1 to C.5 illustrate the average accuracy achieved using the developed methods for real-time feature selection and concept drift detection. Regarding *FIFO*'s size, the same results as the previous appendix (Appendix B) have been achieved. Regarding *Percentage Difference* of Information Gain, it can be seen that lower average accuracies have been recorded in some cases when *Percentage Difference* of Information Gain greater than 50%. Whereas, higher average accuracies have been achieved when *Percentage Difference* of Information Gain between 10% and 50%, and only one exception with *CoverType* dataset.

Table C.1: Summary of average accuracy achieved using uncontrolled real dataset (*CoverType*) with different values (from 10% to 100%) of *Percentage Difference* of Information Gain in combination with different *FIFO*'s sizes (100, 500, and 1000).

Percentage Difference Rate / ***FIFO*'s Size**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**100%**
100	79.35	79.35	79.35	79.35	79.35	79.35	79.35	79.35	79.35	79.35
500	79.32	79.32	79.32	79.32	79.32	79.32	79.32	79.32	79.32	79.32
1000	79.32	79.32	79.32	79.32	79.32	79.32	79.32	79.32	79.32	79.32

Table C.2: Summary of average accuracy achieved using uncontrolled real dataset (*Diabetic Retinopathy Debrecen*) with different values (from 10% to 100%) of *Percentage Difference* of Information Gain in combination with different *FIFO*'s sizes (100, 500, and 1000).

Percentage Difference Rate / ***FIFO*'s Size**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**100%**
100	51.04	50.96	50.96	50.96	50.96	48.87	49.48	49.39	49.39	49.13
500	**52.09**	52.00	51.74	50.61	51.04	49.48	49.39	49.39	48.52	48.52
1000	**52.09**	52.00	51.74	50.61	51.04	49.48	49.39	49.39	48.52	48.52

Table C.3: Summary of average accuracy achieved using uncontrolled real dataset (*Gesture Phase Segmentation*) with different values (from 10% to 100%) of *Percentage Difference* of Information Gain in combination with different *FIFO*'s sizes (100, 500, and 1000).

Percentage Difference Rate / ***FIFO*'s Size**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**100%**
100	72.13	72.13	72.13	80.23	80.23	73.79	73.79	73.79	73.79	73.79
500	**80.63**	80.23	80.23	80.23	80.23	80.23	73.79	73.79	73.79	73.79
1000	**80.63**	80.23	80.23	80.23	80.23	80.23	73.79	73.79	73.79	73.79

Table C.4: Summary of average accuracy achieved using uncontrolled real dataset (*Statlog (Landsat Satellite)*) with different values (from 10% to 100%) of *Percentage Difference* of Information Gain in combination with different *FIFO*'s sizes (100, 500, and 1000).

Percentage Difference Rate / ***FIFO*'s Size**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**100%**
100	81.85	81.63	81.63	81.63	81.38	81.54	81.54	81.54	81.54	81.54
500	81.74	81.60	81.60	**81.94**	81.63	81.63	81.63	81.63	81.63	81.63
1000	81.74	81.60	81.60	**81.94**	81.63	81.63	81.63	81.63	81.63	81.63

Table C.5: Summary of average accuracy achieved using uncontrolled real dataset (*Waveform (with noise)*) with different values (from 10% to 100%) of *Percentage Difference* of Information Gain in combination with different *FIFO*'s sizes (100, 500, and 1000).

Percentage Difference Rate / ***FIFO*'s Size**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**100%**
100	79.30	78.62	78.62	78.62	78.62	78.62	78.62	78.62	78.62	78.62
500	79.82	79.82	**80.68**	**80.68**	**80.68**	**80.68**	**80.68**	**80.68**	**80.68**	**80.68**
1000	79.82	79.82	**80.68**	**80.68**	**80.68**	**80.68**	**80.68**	**80.68**	**80.68**	**80.68**

Appendix D

Error Threshold for Splitting a Micro-Cluster

In this appendix, different values of Θ (error threshold) have been selected randomly (100, 300, 500, 700, and 900) and used for the purpose of splitting Micro-Clusters. MC-NN splits a Micro-Cluster into two new clusters once the error counts ε reaches Θ. Therefore, in this appendix, different values (from 10% to 100%) of *Percentage Difference* of *Split* and *Death* rates in combination with Θ were used. For the experiments in this appendix, the α rate of *LPF* was set to 50% as it yielded good results in most cases as examined in the previous appendices. The new MC-NN with *IQR* was used.

D.1 Artificial and Controlled Real Datasets

Table D.1 illustrates number of concept drifts detected correctly. For the experiments the default parameters stated in Table 3.2 (artificial datasets) and Table 3.3 (controlled real datasets with 6 features) were used unless stated otherwise. The new MC-NN with *IQR* was used. As there are 25 experiments, there is a total of 25 concept drifts to be detected. Where 15 concepts drifts with artificial datasets, and 10 concepts drifts with controlled datasets. Although high *true positive* detections achieved in some cases (18 *false positives* with 7 *true positives*), a relevant value (Θ) that yielded good results for each individual dataset has been identified after re-conducting each dataset using different error counters (between 1 and a window size). Where a window size equals to 10% of number of instances. The relevant values of Θ for datasets are shown in Tables 3.2 and 3.3.

Table D.1: Summary of concept drifts adaptation experiments using artificial and controlled real dataset with different values (from 10% to 100%) of *Percentage Difference* of *Split* and *Death* rates in combination with Θ. The results are reported for the Time 6 which is the time at which features were swapped. Where a first number refers to number of *false positive* detections, whereas a second number (between round brackets) refers to number of *true positive* detections.

Θ \ Percentage Difference Rate	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
100	51(6)	47(5)	29(1)	34(1)	34(1)	31(1)	25(2)	25(2)	25(2)	23(1)
300	32(7)	28(7)	27(7)	23(6)	22(7)	22(7)	20(7)	20(7)	**18(7)**	**18(7)**
500	24(0)	23(0)	22(0)	20(0)	18(0)	18(0)	18(0)	16(0)	16(0)	16(0)
700	16(4)	9(4)	8(4)	8(4)	8(4)	7(4)	7(4)	7(4)	5(4)	5(4)
900	13(0)	11(0)	11(0)	11(0)	11(0)	10(0)	9(0)	7(0)	7(0)	7(0)

D.2 Uncontrolled Real Datasets

Tables D.2 to D.6 illustrate average accuracy achieved using uncontrolled real datasets with the developed methods for real-time feature selection and concept drift detection. For the experiments the default parameters stated in Table 3.4 (uncontrolled real datasets) were used unless stated otherwise. The new MC-NN with *IQR* was used. Hoeffding Tree classifier was used as well. Although high average accuracies achieved in some cases, a relevant value (Θ) that yielded good results for each individual dataset has been identified after re-conducting each dataset using different error counters (between 1 and window size). Where a window size equals to 10% of a number of instances. The relevant values of Θ for datasets are shown in Tables 3.4.

Table D.2: Summary of average accuracy achieved using uncontrolled real dataset (*CoverType*) with different values (from 10% to 100%) of *Percentage Difference* of *Split* and *Death* rates in combination with Θ.

Θ \ Percentage Difference Rate	10%	20%	30%	40%	50%	60%	70%	80%	90%	100%
100	71.63	71.84	71.84	71.84	73.54	73.54	73.54	76.71	76.71	76.71
300	74.62	74.62	75.05	75.05	75.05	74.85	74.85	74.85	74.85	74.85
500	77.52	77.05	76.82	76.79	76.79	76.79	76.79	76.79	76.79	76.79
700	75.57	75.57	75.57	75.57	74.59	74.08	74.08	74.08	74.08	74.08
900	**78.90**	**78.90**	76.82	76.82	76.82	76.79	76.79	76.79	76.79	76.79

Table D.3: Summary of average accuracy achieved using uncontrolled real dataset (*Diabetic Retinopathy Debrecen*) with different values (from 10% to 100%) of *Percentage Difference* of *Split* and *Death* rates in combination with Θ.

Θ \ **Percentage Difference Rate**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**100%**
100	50.09	50.09	50.09	50.09	50.09	50.09	50.09	50.09	50.09	50.09
300	50.09	50.09	50.09	50.09	50.09	50.09	50.09	50.09	50.09	50.09
500	50.09	50.09	50.09	50.09	50.09	50.09	50.09	50.09	50.09	50.09
700	50.09	50.09	50.09	50.09	50.09	50.09	50.09	50.09	50.09	50.09
900	50.09	50.09	50.09	50.09	50.09	50.09	50.09	50.09	50.09	50.09

Table D.4: Summary of average accuracy achieved using uncontrolled real dataset (*Gesture Phase Segmentation*) with different values (from 10% to 100%) of *Percentage Difference* of *Split* and *Death* rates in combination with Θ.

Θ \ **Percentage Difference Rate**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**100%**
100	**80.00**	**80.00**	**80.00**	60.80	60.80	60.80	60.80	54.71	54.71	54.71
300	68.51	68.51	54.71	54.71	54.71	54.71	54.71	54.71	54.71	54.71
500	54.71	54.71	54.71	54.71	54.71	54.71	54.71	54.71	54.71	54.71
700	54.71	54.71	54.71	54.71	54.71	54.71	54.71	54.71	54.71	54.71
900	54.71	54.71	54.71	54.71	54.71	54.71	54.71	54.71	54.71	54.71

Table D.5: Summary of average accuracy achieved using uncontrolled real dataset (*Statlog (Landsat Satellite)*) with different values (from 10% to 100%) of *Percentage Difference* of *Split* and *Death* rates in combination with Θ.

Θ \ **Percentage Difference Rate**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**100%**
100	**81.00**	**81.00**	80.43	80.43	80.43	80.43	80.43	77.63	77.63	77.63
300	80.61	80.61	80.61	80.61	80.61	80.61	76.28	76.28	76.28	76.28
500	78.49	78.49	78.49	78.49	76.28	76.28	76.28	76.28	76.28	76.28
700	76.28	76.28	76.28	76.28	76.28	76.28	76.28	76.28	76.28	76.28
900	76.28	76.28	76.28	76.28	76.28	76.28	76.28	76.28	76.28	76.28

Table D.6: Summary of average accuracy achieved using uncontrolled real dataset (*Waveform (with noise)*) with different values (from 10% to 100%) of *Percentage Difference* of *Split* and *Death* rates in combination with Θ.

Θ \ **Percentage Difference Rate**	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**100%**
100	79.60	79.60	79.46	79.46	79.46	79.46	79.46	79.46	79.46	79.46
300	71.62	76.60	**80.42**	**80.42**	**80.42**	**80.42**	**80.42**	**80.42**	**80.42**	**80.42**
500	76.22	76.22	**80.42**	**80.42**	**80.42**	**80.42**	**80.42**	**80.42**	**80.42**	**80.42**
700	**80.42**	**80.42**	**80.42**	**80.42**	**80.42**	**80.42**	**80.42**	**80.42**	**80.42**	**80.42**
900	**80.42**	**80.42**	**80.42**	**80.42**	**80.42**	**80.42**	**80.42**	**80.42**	**80.42**	**80.42**

Appendix E

Percentage Number of Features with Maximum Velocity Combined with Variance or IQR

In this appendix, different values of percentage number of features with maximum *Velocity* combined with *Variance* or *IQR* were used for the purpose of feature analysis and feature selection using artificial and controlled real datasets. For the experiments in this appendix, the α rate of *LPF* and *Percentage Difference* of *Split* and *Death* rates were set to 50% as it yielded good results in most cases (Appendix A).

E.1 Artificial Datasets

Table E.1 illustrates number of features tracked with maximum *Velocity* combined with *Variance*. For the experiments the default parameters stated in Table 3.2 were used unless stated otherwise. In order to identify a best highest percentage number of features with maximum *Velocity* combined with *Variance*, 3 percentage numbers are stated in the table are 25%, 50%, and 75%. Where number of *true positive* and *false positive* is recorded for each percentage number in the table. As there are 3 features for each dataset, It is possible that 25% retrieves only 1 feature. Whereas, 50% and 75% retrieve 2 features.

In Table E.1, numbers indicated with a * indicate potentially *false positives* detection of features that changed their relevance. Potentially means in this case that there was also an unexpected concept drift. However, it is not possible to verify with absolute certainty if the detection was indeed a *false positive*. Whereas, numbers indicated with a (-) reflect a drift was not detected. It can be seen that the 50% and 75% achieved a much higher *true positive* detection of features that changed.

Table E.1: Summary of the experimental results with artificial datasets. 3 percentage numbers are stated in the table which are 25%, 50%, and 75%. These percentages represent the highest percentage number of features with maximum *Velocity* combined with *Variance*. The original MC-NN with *Variance* was used. The results are reported for the Time 6 which is the time of swapped features.

Dataset	25% (*true positive*)	25% (*false positive*)	50% (*true positive*)	50% (*false positive*)	75% (*true positive*)	75% (*false positive*)
SEA with Noise 0%	1	1	2		2	
SEA with Noise 5%	1		1		1	
SEA with Noise 15%			1		1	
SEA with Noise 25%			1		1	
SEA with Noise 35%			1		1	
HyperPlane with Noise 0%		1		1		1
HyperPlane with Noise 5%		1		1		1
HyperPlane with Noise 15%				1		1
HyperPlane with Noise 25%						
HyperPlane with Noise 35%		2		1		1
RandomTree with Noise 0%						
RandomTree with Noise 5%			1		1	
RandomTree with Noise 15%			1		1	
RandomTree with Noise 25%				1*		1*
RandomTree with Noise 35%		1(-)		1(-)		1(-)
Total	2	6	8	6	8	6

Table E.2 illustrates number of features tracked with maximum *Velocity* combined with *IQR*. As mentioned before, for the experiments the default parameters of artificial datasets stated in Table 3.2 were used unless stated otherwise. It can be seen that the 50% and 75% achieved a much higher *true positive* detection of features that changed.

Table E.2: Summary of the experimental results with artificial datasets. 3 percentage numbers are stated in the table which are 25%, 50%, and 75%. These percentages represent the highest percentage number of features with maximum *Velocity* combined with *IQR*. The new MC-NN with *IQR* was used. The results are reported for the Time 6 which is the time of swapped features.

Dataset	25% (*true positive*)	25% (*false positive*)	50% (*true positive*)	50% (*false positive*)	75% (*true positive*)	75% (*false positive*)
SEA with Noise 0%	1		2		2	
SEA with Noise 5%	1		2		2	
SEA with Noise 15%	1		1		1	
SEA with Noise 25%	1		1		1	
SEA with Noise 35%	1		2		2	
HyperPlane with Noise 0%	1		1		1	
HyperPlane with Noise 5%	1		1		1	
HyperPlane with Noise 15%	1		1		1	
HyperPlane with Noise 25%		1	1	1	1	1
HyperPlane with Noise 35%	1		1	1	1	1
RandomTree with Noise 0%	1		2		2	
RandomTree with Noise 5%	1		1		1	
RandomTree with Noise 15%	1		2		2	
RandomTree with Noise 25%		1 + 1*	1	1 + 2*	1	1 + 2*
RandomTree with Noise 35%		1	1	1	1	1
Total	12	4	20	6	20	6

E.2 Controlled Real Datasets with 6 Features

Table E.3 illustrates number of features tracked with maximum *Velocity* combined with *Variance*. For the experiments the default parameters stated in Table 3.3 were used unless stated otherwise. In order to identify a best highest percentage number of features with maximum *Velocity* combined with *Variance*, 3 percentage numbers are stated in the table which are 25%, 50%, and 75%. Where number of *true positive* and *false positive* is recorded for each percentage number in the table. As there are 6 features for each dataset, it is possible that 25% retrieves only 2 features. 50% retrieves 3 features. Whereas, 75% retrieves 5 features. It can be seen that the 50% and 75% achieved a much higher *true positive* detection of features that changed.

Table E.3: Summary of the experimental results with controlled real datasets. 3 percentage numbers are stated in the table which are 25%, 50%, and 75%. These percentages represent the highest percentage number of features with maximum *Velocity* combined with *Variance*. The original MC-NN with *Variance* was used. The results are reported for the Time 6 which is the time of swapped features.

Dataset	25% (*true positive*)	25% (*false positive*)	50% (*true positive*)	50% (*false positive*)	75% (*true positive*)	75% (*false positive*)
CoverType (2 features swapped)		1(-)		2(-)		2(-)
CoverType (4 features swapped)		1(-)		1(-)		1(-)
Diabetic Retinopathy Debrecen (2 features swapped)						
Diabetic Retinopathy Debrecen (4 features swapped)		1(-)		1(-)		1(-)
Gesture Phase Segmentation (2 features swapped)						
Gesture Phase Segmentation (4 features swapped)		1(-)		1(-)		1(-)
Statlog (Landsat Satellite) (2 features swapped)	1		1		1	
Statlog (Landsat Satellite) (4 features swapped)						
Waveform (with Noise) (2 features swapped)						
Waveform (with Noise) (4 features swapped)						
Total	1	4	1	5	1	5

Table E.4 illustrates number of features tracked with maximum *Velocity* combined with *IQR*. As mentioned before, for the experiments the default parameters of controlled real datasets stated in Table 3.3 were used unless stated otherwise. It can be seen that the 50% and 75% achieved a much higher *true positive* detection of features that changed.

Table E.4: Summary of the experimental results with controlled real datasets. 3 percentage numbers are stated in the table which are 25%, 50%, and 75%. These percentages represent the highest percentage number of features with maximum *Velocity* combined with *IQR*. The new MC-NN with *IQR* was used. The results are reported for the Time 6 which is the time of swapped features.

Dataset	25% (*true positive*)	25% (*false positive*)	50% (*true positive*)	50% (*false positive*)	75% (*true positive*)	75% (*false positive*)
CoverType (2 features swapped)	1	1	2	1	2	1
CoverType (4 features swapped)	1	1	2	1	2	1
Diabetic Retinopathy Debrecen (2 features swapped)	1		1	1	1	1
Diabetic Retinopathy Debrecen (4 features swapped)	2		2	1	2	1
Gesture Phase Segmentation (2 features swapped)	1		2		2	
Gesture Phase Segmentation (4 features swapped)	1		2		2	
Statlog (Landsat Satellite) (2 features swapped)	1	1*	1	1*	1	1*
Statlog (Landsat Satellite) (4 features swapped)	1		1		1	
Waveform (with Noise) (2 features swapped)	1		1		1	
Waveform (with Noise) (4 features swapped)	2		3		3	
Total	12	3	17	5	17	5

E.3 Uncontrolled Real Datasets

Table E.5 illustrates average accuracy achieved of the Hoeffding Tree classifier using the developed real-time feature selection method. For the experiments the default parameters stated in Table 3.4 were used unless stated otherwise. In order to identify a best highest percentage number of features with maximum *Velocity* combined with *IQR*, 3 percentage numbers are stated in the table which are 25%, 50%, and 75%. Regarding *CoverType* dataset, the same results have been achieved with all percentage numbers. Regarding *Diabetic Retinopathy Debrecen* dataset, the best average accuracy has achieved with 50%. Regarding *Gesture Phase Segmentation* dataset, the same results have been achieved with 50% and 75%. Regarding *Statlog (Landsat Satellite)* dataset, the best average accuracy has achieved with 75%. Regarding *Waveform (with Noise)* dataset, the same results have been achieved with 50% and 75%.

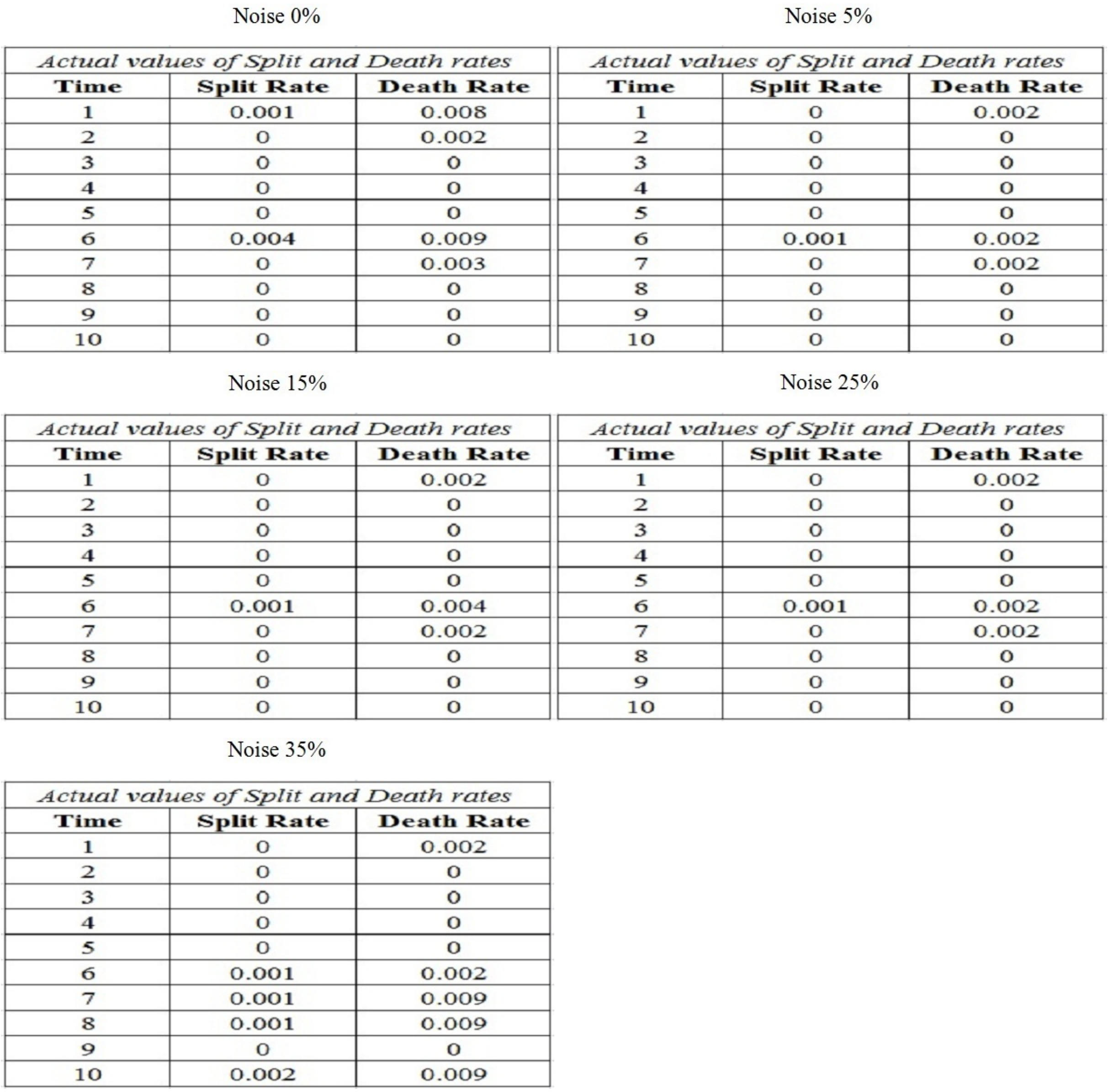
Table E.5: Summary of the experimental results with uncontrolled real datasets. 3 percentage numbers are stated in the table which are 25%, 50%, and 75%. These percentages represent the highest percentage number of features with maximum *Velocity* combined with *IQR*. The results are reported for the average accuracy of the Hoeffding Tree classifier achieved using the developed real-time feature selection method.

Dataset	25%	50%	75%
CoverType	79.32	79.32	79.32
Diabetic Retinopathy Debrecen	50.26	**51.04**	50.35
Gesture Phase Segmentation	73.16	**80.23**	**80.23**
Statlog (Landsat Satellite)	81.47	81.63	**81.92**
Waveform (with Noise)	80.26	**80.68**	**80.68**

Appendix F

Actual Values of Split and Death Rates

This appendix presents the actual values of *Split* and *Death* rates which were used for the purpose of concept drift detection using artificial and controlled real datasets.



Noise 0%

Actual values of Split and Death rates		
Time	**Split Rate**	**Death Rate**
1	0.001	0.008
2	0	0.002
3	0	0
4	0	0
5	0	0
6	0.004	0.009
7	0	0.003
8	0	0
9	0	0
10	0	0

Noise 5%

Actual values of Split and Death rates		
Time	**Split Rate**	**Death Rate**
1	0	0.002
2	0	0
3	0	0
4	0	0
5	0	0
6	0.001	0.002
7	0	0.002
8	0	0
9	0	0
10	0	0

Noise 15%

Actual values of Split and Death rates		
Time	**Split Rate**	**Death Rate**
1	0	0.002
2	0	0
3	0	0
4	0	0
5	0	0
6	0.001	0.004
7	0	0.002
8	0	0
9	0	0
10	0	0

Noise 25%

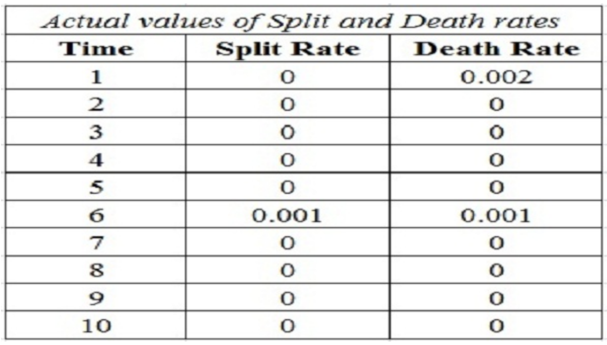
Actual values of Split and Death rates		
Time	**Split Rate**	**Death Rate**
1	0	0.002
2	0	0
3	0	0
4	0	0
5	0	0
6	0.001	0.002
7	0	0.002
8	0	0
9	0	0
10	0	0

Noise 35%

Actual values of Split and Death rates		
Time	**Split Rate**	**Death Rate**
1	0	0.002
2	0	0
3	0	0
4	0	0
5	0	0
6	0.001	0.002
7	0.001	0.009
8	0.001	0.009
9	0	0
10	0.002	0.009

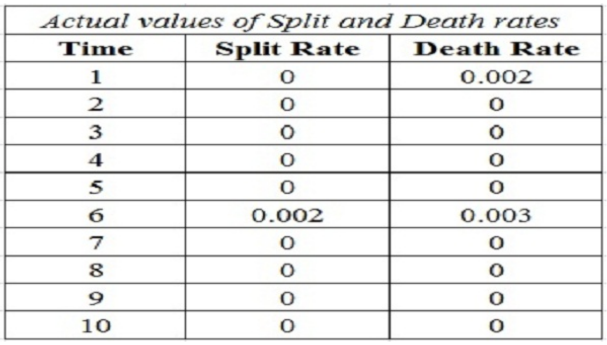
Figure F.1: The actual values of *Split* and *Death* rates of *SEA* data stream generator for drift detection.

Noise 0%



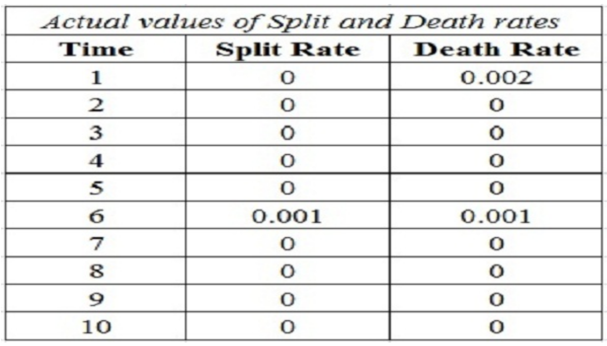
Actual values of Split and Death rates		
Time	**Split Rate**	**Death Rate**
1	0	0.002
2	0	0
3	0	0
4	0	0
5	0	0
6	0.001	0.001
7	0	0
8	0	0
9	0	0
10	0	0

Noise 5%



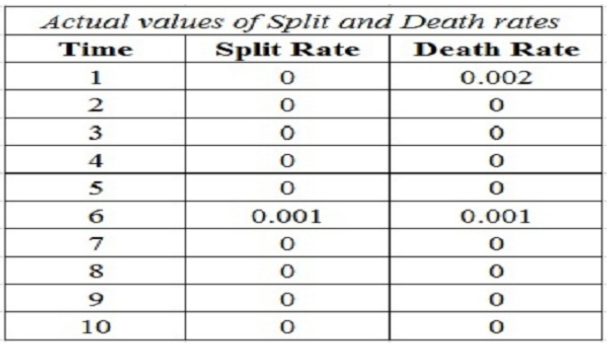
Actual values of Split and Death rates		
Time	**Split Rate**	**Death Rate**
1	0	0.002
2	0	0
3	0	0
4	0	0
5	0	0
6	0.002	0.003
7	0	0
8	0	0
9	0	0
10	0	0

Noise 15%



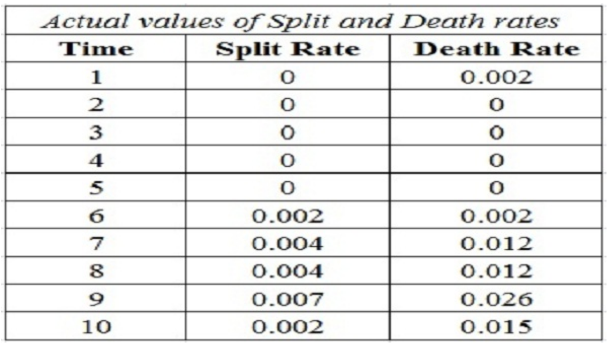
Actual values of Split and Death rates		
Time	**Split Rate**	**Death Rate**
1	0	0.002
2	0	0
3	0	0
4	0	0
5	0	0
6	0.001	0.001
7	0	0
8	0	0
9	0	0
10	0	0

Noise 25%



Actual values of Split and Death rates		
Time	**Split Rate**	**Death Rate**
1	0	0.002
2	0	0
3	0	0
4	0	0
5	0	0
6	0.001	0.001
7	0	0
8	0	0
9	0	0
10	0	0

Noise 35%



Actual values of Split and Death rates		
Time	**Split Rate**	**Death Rate**
1	0	0.002
2	0	0
3	0	0
4	0	0
5	0	0
6	0.002	0.002
7	0.004	0.012
8	0.004	0.012
9	0.007	0.026
10	0.002	0.015

Figure F.2: The actual values of *Split* and *Death* rates of *HyperPlane* data stream generator for drift detection.

Noise 0%

Actual values of Split and Death rates		
Time	**Split Rate**	**Death Rate**
1	0	0.001
2	0	0
3	0	0
4	0	0
5	0	0
6	0.001	0.003
7	0	0.001
8	0	0
9	0	0
10	0	0

Noise 5%

Actual values of Split and Death rates		
Time	**Split Rate**	**Death Rate**
1	0	0.001
2	0	0
3	0	0
4	0	0
5	0	0
6	0.001	0.001
7	0	0.002
8	0	0
9	0	0
10	0	0

Noise 15%

Actual values of Split and Death rates		
Time	**Split Rate**	**Death Rate**
1	0	0.001
2	0	0
3	0	0
4	0	0
5	0	0
6	0.001	0.001
7	0	0.001
8	0	0
9	0	0
10	0	0

Noise 25%

Actual values of Split and Death rates		
Time	**Split Rate**	**Death Rate**
1	0	0.001
2	0	0
3	0	0
4	0	0
5	0	0
6	0.001	0.001
7	0	0.001
8	0	0
9	0	0
10	0.001	0.001

Noise 35%

Actual values of Split and Death rates		
Time	**Split Rate**	**Death Rate**
1	0	0.001
2	0	0
3	0.001	0.001
4	0.001	0.01
5	0	0
6	0.001	0.001
7	0	0.002
8	0.002	0.003
9	0.001	0.001
10	0.001	0.001

Figure F.3: The actual values of *Split* and *Death* rates of *Random Tree* data stream generator for drift detection.

2 Features Swapped

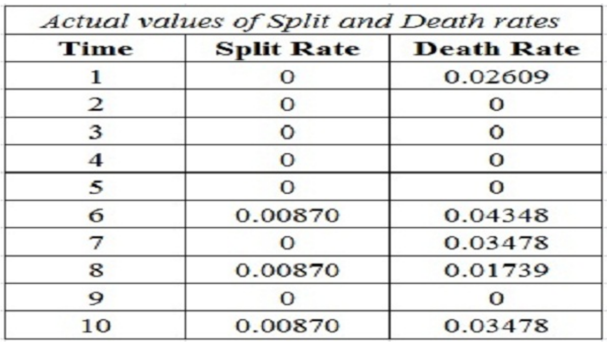
Actual values of Split and Death rates		
Time	**Split Rate**	**Death Rate**
1	0.00012	0.00038
2	0.00014	0.00033
3	0.00012	0.00024
4	0.00010	0.00026
5	0.00015	0.00038
6	0.00022	0.00043
7	0.00015	0.00031
8	0.00014	0.00024
9	0.00012	0.00026
10	0.00012	0.00028

4 Features Swapped

Actual values of Split and Death rates		
Time	**Split Rate**	**Death Rate**
1	0.00000	0.00017
2	0.00002	0.00003
3	0.00003	0.00007
4	0.00002	0.00014
5	0.00002	0.00005
6	0.00005	0.00010
7	0.00002	0.00015
8	0.00007	0.00012
9	0.00002	0.00003
10	0.00003	0.00009

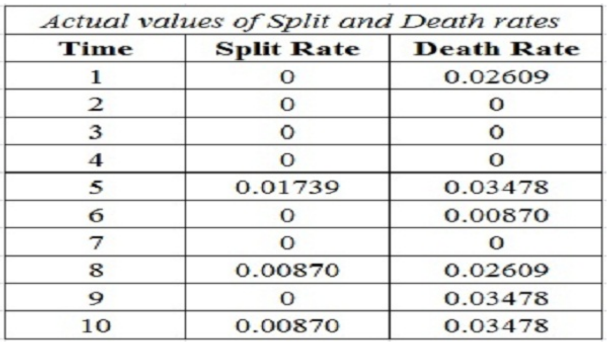
Figure F.4: The actual values of *Split* and *Death* rates of *CoverType* dataset with 6 features for drift detection.

2 Features Swapped



Actual values of Split and Death rates		
Time	**Split Rate**	**Death Rate**
1	0	0.02609
2	0	0
3	0	0
4	0	0
5	0	0
6	0.00870	0.04348
7	0	0.03478
8	0.00870	0.01739
9	0	0
10	0.00870	0.03478

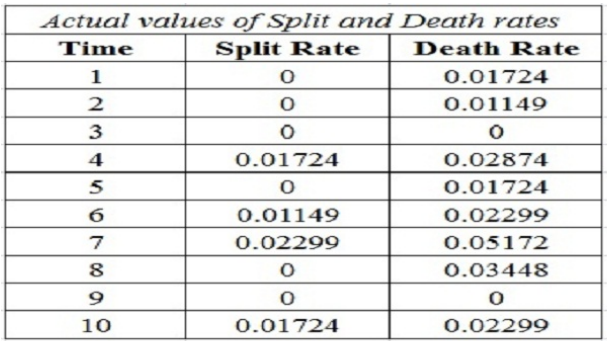
4 Features Swapped



Actual values of Split and Death rates		
Time	**Split Rate**	**Death Rate**
1	0	0.02609
2	0	0
3	0	0
4	0	0
5	0.01739	0.03478
6	0	0.00870
7	0	0
8	0.00870	0.02609
9	0	0.03478
10	0.00870	0.03478

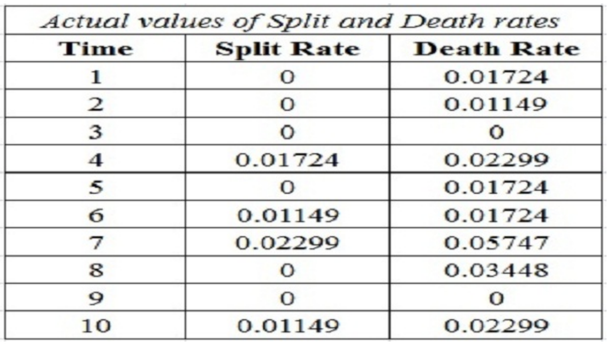
Figure F.5: The actual values of *Split* and *Death* rates of *Diabetic Retinopathy Debrecen* dataset with 6 features for drift detection.

2 Features Swapped



Actual values of Split and Death rates		
Time	**Split Rate**	**Death Rate**
1	0	0.01724
2	0	0.01149
3	0	0
4	0.01724	0.02874
5	0	0.01724
6	0.01149	0.02299
7	0.02299	0.05172
8	0	0.03448
9	0	0
10	0.01724	0.02299

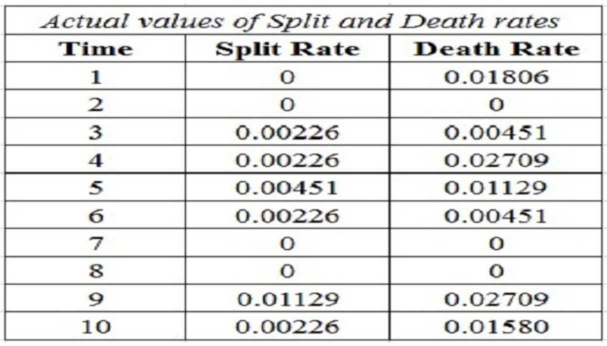
4 Features Swapped



Actual values of Split and Death rates		
Time	**Split Rate**	**Death Rate**
1	0	0.01724
2	0	0.01149
3	0	0
4	0.01724	0.02299
5	0	0.01724
6	0.01149	0.01724
7	0.02299	0.05747
8	0	0.03448
9	0	0
10	0.01149	0.02299

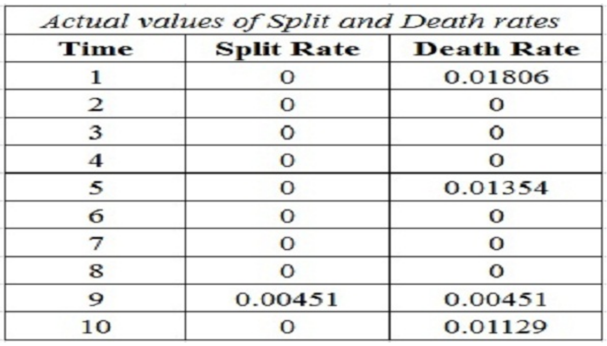
Figure F.6: The actual values of *Split* and *Death* rates of *Gesture Phase Segmentation* dataset with 6 features for drift detection.

2 Features Swapped



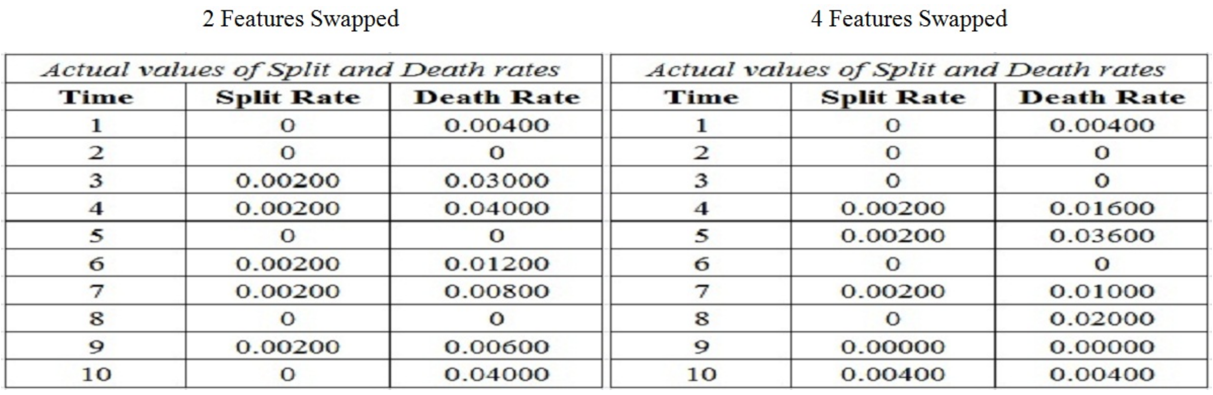
Actual values of Split and Death rates		
Time	**Split Rate**	**Death Rate**
1	0	0.01806
2	0	0
3	0.00226	0.00451
4	0.00226	0.02709
5	0.00451	0.01129
6	0.00226	0.00451
7	0	0
8	0	0
9	0.01129	0.02709
10	0.00226	0.01580

4 Features Swapped



Actual values of Split and Death rates		
Time	**Split Rate**	**Death Rate**
1	0	0.01806
2	0	0
3	0	0
4	0	0
5	0	0.01354
6	0	0
7	0	0
8	0	0
9	0.00451	0.00451
10	0	0.01129

Figure F.7: The actual values of *Split* and *Death* rates of *Statlog (Landsat Satellite)* dataset with 6 features for drift detection.


2 Features Swapped

Actual values of Split and Death rates		
Time	**Split Rate**	**Death Rate**
1	0	0.00400
2	0	0
3	0.00200	0.03000
4	0.00200	0.04000
5	0	0
6	0.00200	0.01200
7	0.00200	0.00800
8	0	0
9	0.00200	0.00600
10	0	0.04000

4 Features Swapped

Actual values of Split and Death rates		
Time	**Split Rate**	**Death Rate**
1	0	0.00400
2	0	0
3	0	0
4	0.00200	0.01600
5	0.00200	0.03600
6	0	0
7	0.00200	0.01000
8	0	0.02000
9	0.00000	0.00000
10	0.00400	0.00400

Figure F.8: The actual values of *Split* and *Death* rates of *Waveform (with Noise)* dataset with 6 features for drift detection.

Appendix G

The Effects of Feature-Bias, Noise, and Outliers on Velocity of Features

In this appendix, the effects of feature-bias, noise, and outliers on *Velocity* of features are addressed. This appendix applies the original MC-NN with *Variance* and the new MC-NN with *IQR* in combination with Min-Max *Normalisation* and *LPF*. For the experiment, the default parameters stated in Table 3.3 (i.e., controlled real datasets) of the technique were used unless stated otherwise. Figures G.1 to G.10 visualise the results for minimising the effect of feature-bias, noise, and outliers. Part (a) of the figures refers to the method based on *Variance*, and part (b) of the figures refers to the method based on *IQR*. In part (a) of the figures, although real-time Min-Max *Normalisation* and *LPF* were applied, *Velocity* rates are very high due to the effect of outliers which have a significant effect on *Variance* and the detection of both concept drift and the involved features in drifting. Whereas, in part (b) of the figures, *Velocity* rates are cleaner/smoother in comparison with the in part (a) as *IQR* is considered robust to outliers, and then the detection of concept drift can be applied more accurately as well as tracking features using *IQR* (see Section 5.4).

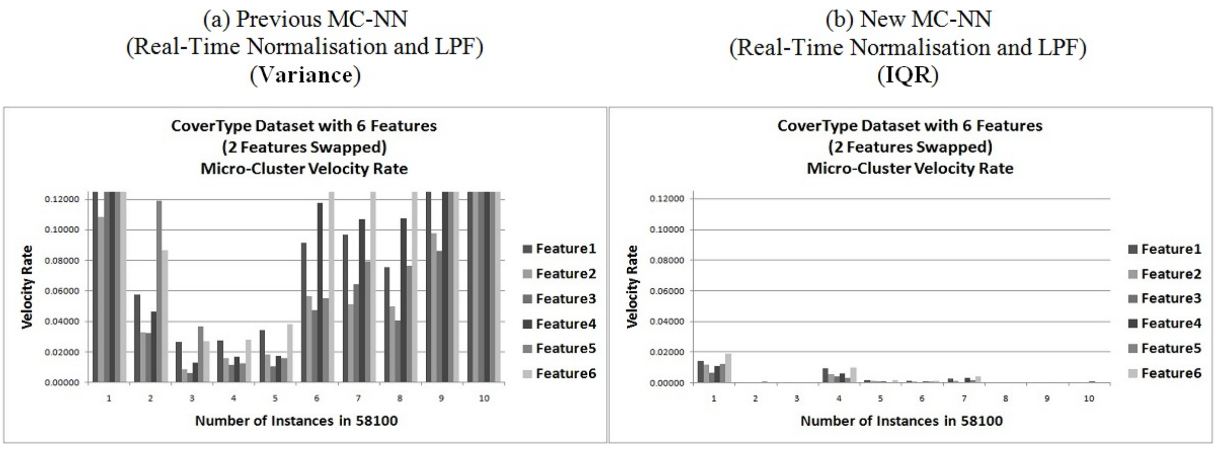


Figure G.1: The results of *CoverType* dataset with 6 features (2 features swapped) using Min-Max, *LPF*, and *IQR* for minimising the effects of feature-bias, noise, and outliers.

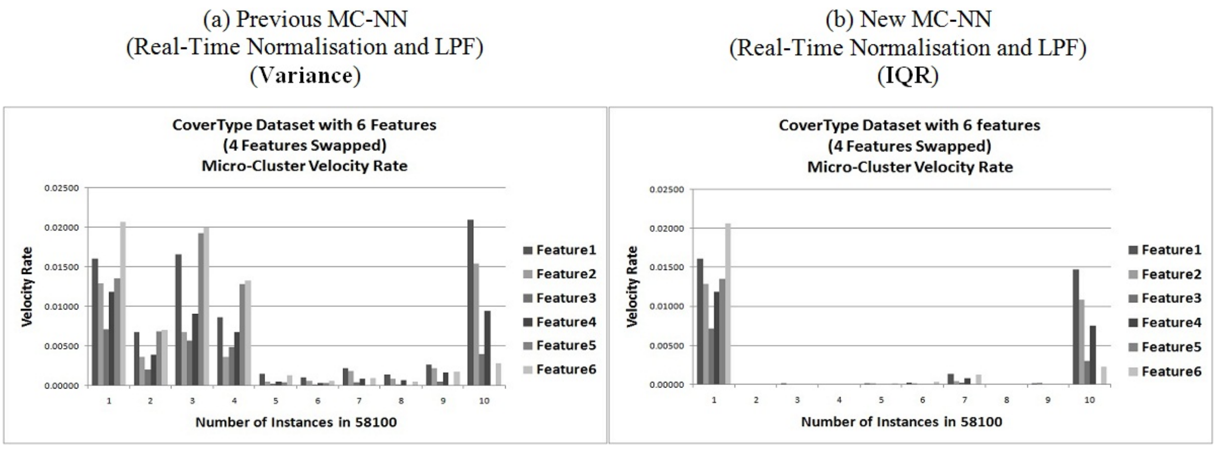


Figure G.2: The results of *CoverType* dataset with 6 features (4 features swapped) using Min-Max, *LPF*, and *IQR* for minimising the effects of feature-bias, noise, and outliers.

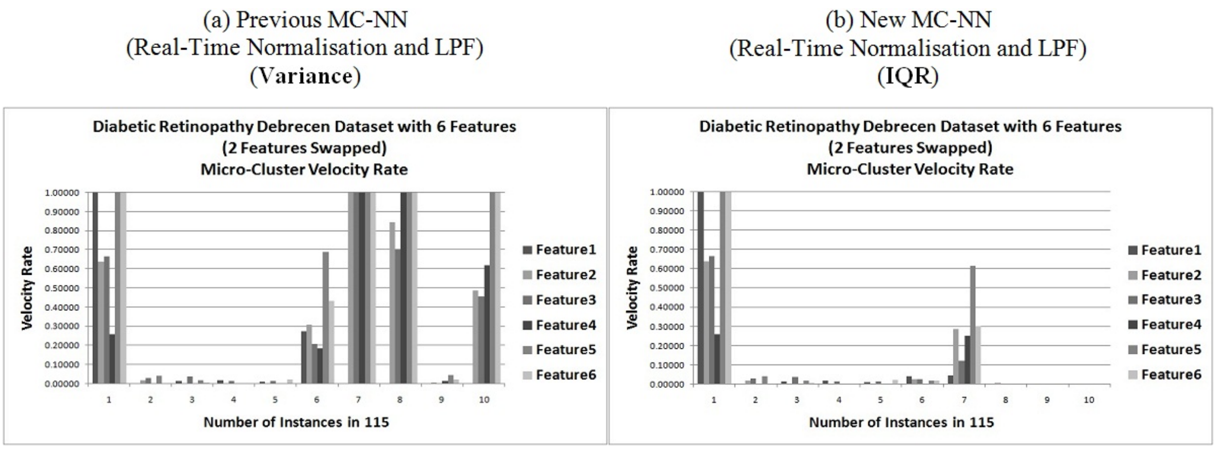


Figure G.3: The results of *Diabetic Retinopathy Debrecen* dataset with 6 features (2 features swapped) using Min-Max, *LPF*, and *IQR* for minimising the effects of feature-bias, noise, and outliers.

(a) Previous MC-NN
(Real-Time Normalisation and LPF)
(Variance)

(b) New MC-NN
(Real-Time Normalisation and LPF)
(IQR)

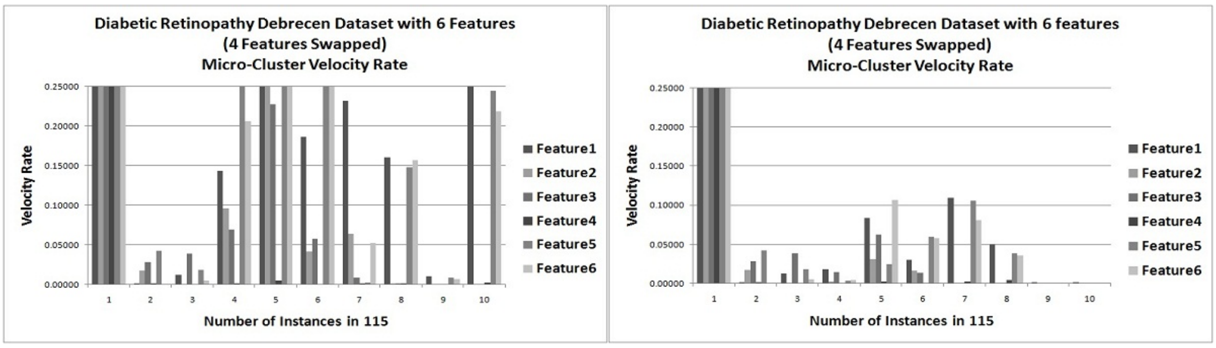


Figure G.4: The results of *Diabetic Retinopathy Debrecen* dataset with 6 features (4 features swapped) using Min-Max, *LPF*, and *IQR* for minimising the effects of feature-bias, noise, and outliers.

(a) Previous MC-NN
(Real-Time Normalisation and LPF)
(Variance)

(b) New MC-NN
(Real-Time Normalisation and LPF)
(IQR)

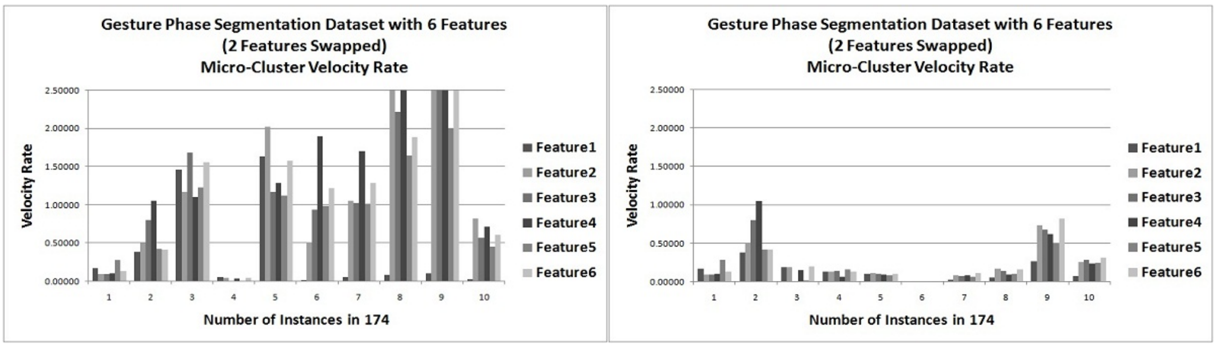


Figure G.5: The results of *Gesture Phase Segmentation* dataset with 6 features (2 features swapped) using Min-Max, *LPF*, and *IQR* for minimising the effects of feature-bias, noise, and outliers.

(a) Previous MC-NN
(Real-Time Normalisation and LPF)
(Variance)

(b) New MC-NN
(Real-Time Normalisation and LPF)
(IQR)

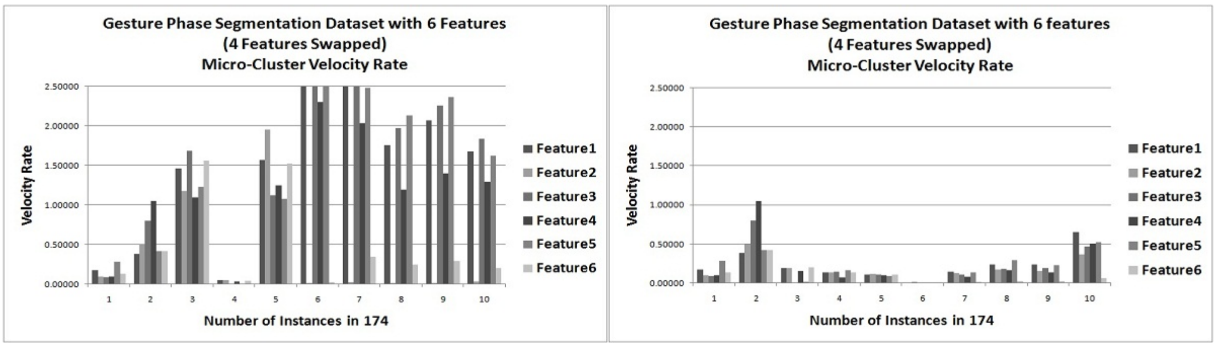


Figure G.6: The results of *Gesture Phase Segmentation* dataset with 6 features (4 features swapped) using Min-Max, *LPF*, and *IQR* for minimising the effects of feature-bias, noise, and outliers.

(a) Previous MC-NN
(Real-Time Normalisation and LPF)
(Variance)

(b) New MC-NN
(Real-Time Normalisation and LPF)
(IQR)

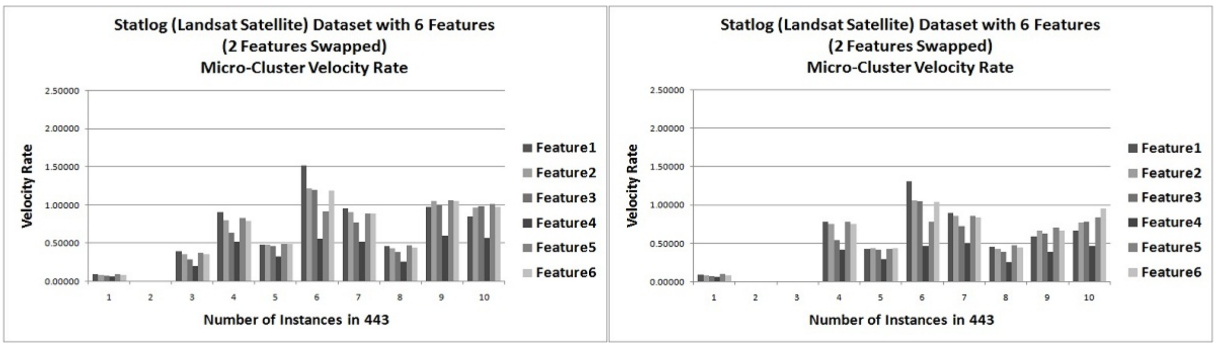


Figure G.7: The results of *Statlog (Landsat Satellite)* dataset with 6 features (2 features swapped) using Min-Max, *LPF*, and *IQR* for minimising the effects of feature-bias, noise, and outliers.

(a) Previous MC-NN
(Real-Time Normalisation and LPF)
(Variance)

(b) New MC-NN
(Real-Time Normalisation and LPF)
(IQR)

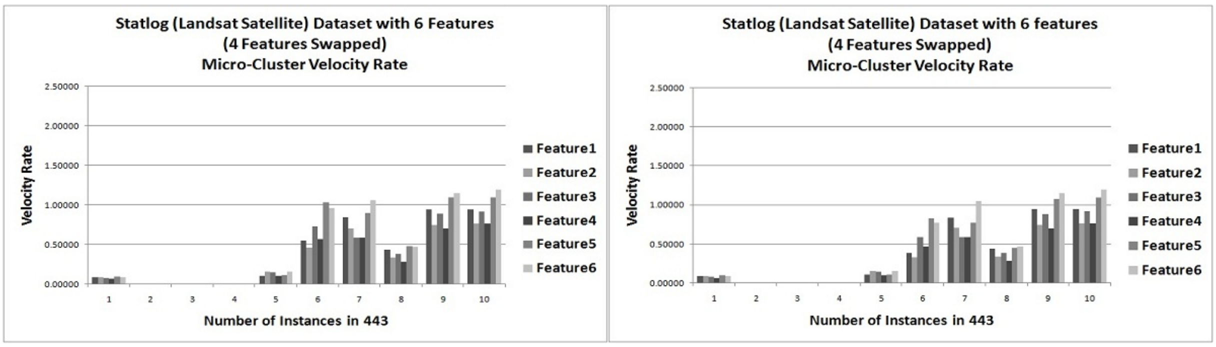


Figure G.8: The results of *Statlog (Landsat Satellite)* dataset with 6 features (4 features swapped) using Min-Max, *LPF*, and *IQR* for minimising the effects of feature-bias, noise, and outliers.

(a) Previous MC-NN
(Real-Time Normalisation and LPF)
(Variance)

(b) New MC-NN
(Real-Time Normalisation and LPF)
(IQR)

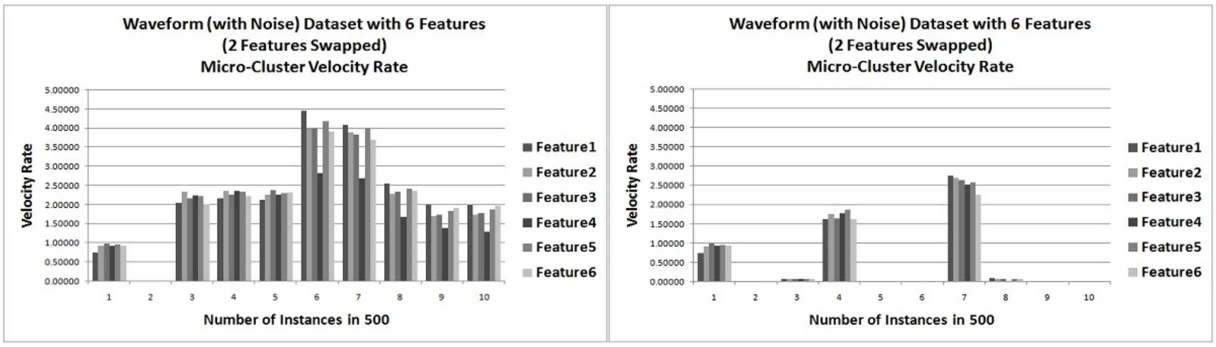


Figure G.9: The results of *Waveform (with Noise)* dataset with 6 features (2 features swapped) using Min-Max, *LPF*, and *IQR* for minimising the effects of feature-bias, noise, and outliers.

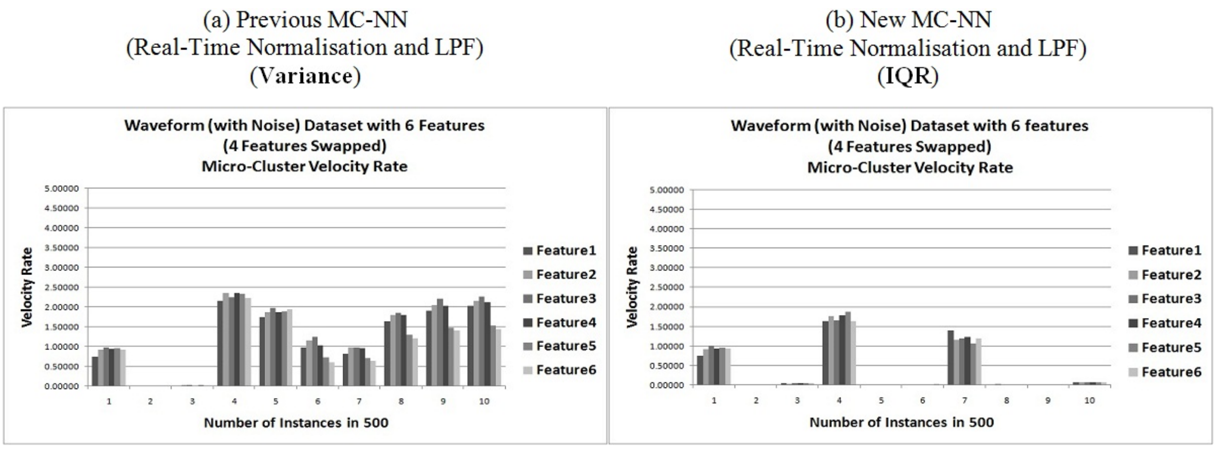


Figure G.10: The results of *Waveform (with Noise)* dataset with 6 features (4 features swapped) using Min-Max, *LPF*, and *IQR* for minimising the effects of feature-bias, noise, and outliers.

Appendix H

Gradual Concept Drift Detection using Artificial Datasets

H.1 Experimental Setup of Artificial Datasets

Three datasets were generated using the MOA data stream generators (*SEA*, *HyperPlane*, and *Random Tree*), each comprising three features, two class labels and a concept drift. The concept drift was always induced halfway through the stream by swapping two features gradually (i.e., induce a *gradual concept drift* as known ground truth by swapping features gradually), in order to show that the developed method is able to detect different types of concept drift. For example, features were swapped from 5000 instances to 5099 instances, 5200 instances to 5299 instances, 5400 instances to 5499 instances, 5600 instances to 5699 instances, 5800 instances to 5899 instances, and from 6000 to 9999 instances (end of the stream). The participation threshold Ω was set to 50 for each experiment, which yielded good results in most cases, as stated in (Tennant et al., 2017). The error threshold Θ was set to the best performing one for each dataset. Table H.1 shows an overview of generated streams including the settings of the developed method and which features have been swapped. In the experiments, a window's size equals to 10% of the total number of instances. Hence, the expression **Time t** refers to a particular time window. I.e. according to the figures in this appendix time *T=1* refers to instances 1-1000, *T=2* to instances 1001-2000, etc.

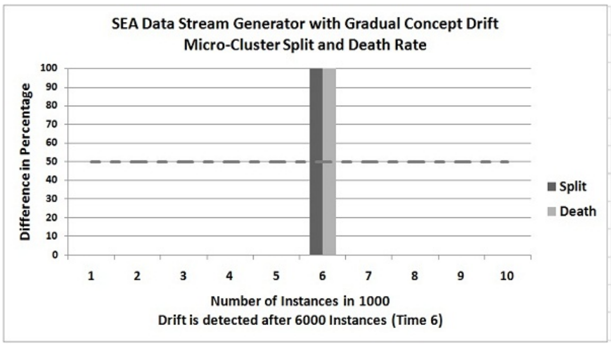
Table H.1: Setup of the artificial datasets. Drifts were generated by swapping features gradually.

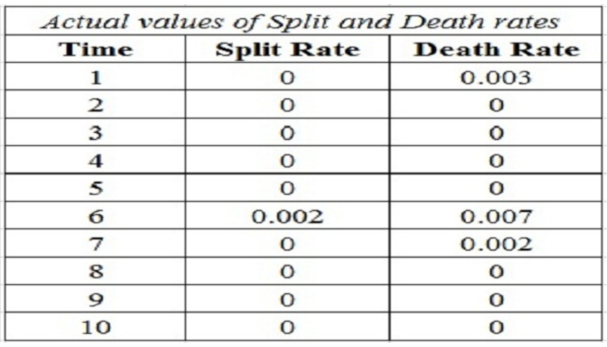
Dataset	Number of Instances Generated	Window Size	Θ	Index of Swapped Features	Start of Swapped Features
SEA	10,000	1000	15	2 with 3	6000
HyperPlane	10,000	1000	2	2 with 3	6000
Random Tree	10,000	1000	359	2 with 3	6000

H.2 Results

H.2.1 Real-Time Concept Detection Method

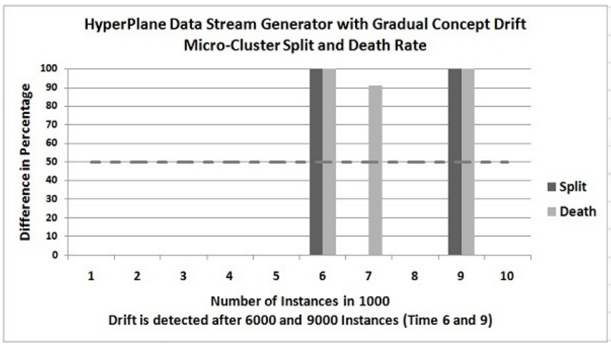
Micro-Clusters' *Split* and *Death* rates were used for detecting drifts. For the experiment the default parameters stated in Table H.1 of the method were used unless stated otherwise.

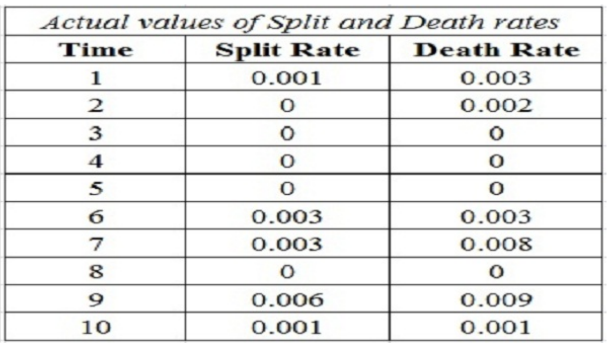




Actual values of Split and Death rates		
Time	**Split Rate**	**Death Rate**
1	0	0.003
2	0	0
3	0	0
4	0	0
5	0	0
6	0.002	0.007
7	0	0.002
8	0	0
9	0	0
10	0	0

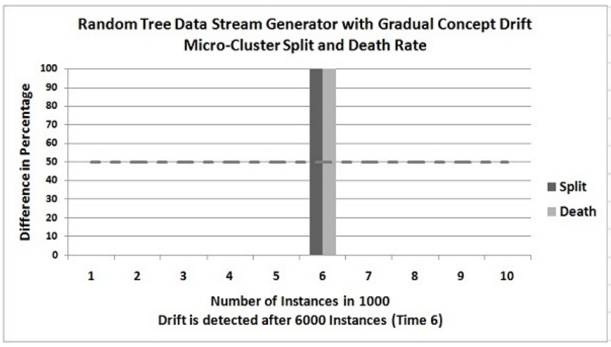
Figure H.1: The results of *SEA* data stream generator using Micro-Cluster *Percentage Difference* of *Split* and *Death* rates for drift detection.

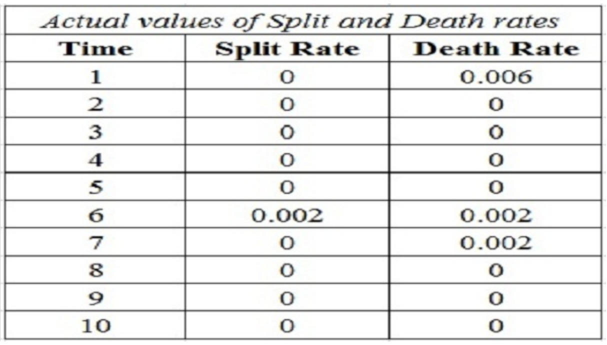




Actual values of Split and Death rates		
Time	**Split Rate**	**Death Rate**
1	0.001	0.003
2	0	0.002
3	0	0
4	0	0
5	0	0
6	0.003	0.003
7	0.003	0.008
8	0	0
9	0.006	0.009
10	0.001	0.001

Figure H.2: The results of *HyperPlane* data stream generator using Micro-Cluster *Percentage Difference* of *Split* and *Death* rates for drift detection.





Actual values of Split and Death rates		
Time	**Split Rate**	**Death Rate**
1	0	0.006
2	0	0
3	0	0
4	0	0
5	0	0
6	0.002	0.002
7	0	0.002
8	0	0
9	0	0
10	0	0

Figure H.3: The results of *Random Tree* data stream generator using Micro-Cluster *Percentage Difference* of *Split* and *Death* rates for drift detection.

In Figures (H.1 to H.3), the focus is given more at the times when concept drift was introduced and features were swapped. During this time as higher *Split* and *Death* rates are expected as the set of Micro-Clusters adapts to the new concept the feature swap. In the figures, it can be seen that the *Split* and *Death* rates at the time of concept drift (after time 5) increase as expected for all artificial data streams. Next, the concept drift detection method was compared with existing state-of-the-art drift detection methods CUSUM, DDM, EDDM, EWMA, and ADWIN (see section 2 for more details about these methods) on the same artificial data streams. Table H.2 shows the time when each of the methods including the developed method, detected a drift and if it was detected on time. As it can be seen, the developed method always detected the drift at the correct time.

Table H.2: Adaptation to concept drift using the initially developed and other state-of-the-art methods.

Generator	Method	Number of Drift Detections	Times when Drift Detected	Drift Detected
SEA	**The Developed Method**	1	6	**Correctly**
	CUSUM	1	5	Incorrectly
	DDM	2	4 and 5	Incorrectly
	EDDM	3	2,3, and 5	Incorrectly
	EWMA	6	2 to 4,6,7, and 9	**Correctly**
	ADWIN	4	5 to 7 and 9	**Correctly**
HyperPlane	**The Developed Method**	2	6 and 9	**Correctly**
	CUSUM	-	-	-
	DDM	1	6	**Correctly**
	EDDM	2	6 and 8	**Correctly**
	EWMA	6	1,2, 4 to 6, and 8	**Correctly**
	ADWIN	2	2 and 6	**Correctly**
RandomTree	**The Developed Method**	1	6	**Correctly**
	CUSUM	1	6	**Correctly**
	DDM	1	6	**Correctly**
	EDDM	1	6	**Correctly**
	EWMA	8	1 to 5 and 7 to 9	Incorrectly
	ADWIN	3	4 and 6	**Correctly**

If a method detects a drift falsely, then this would require unnecessary adaptation by the classifier to a non-existing drift. This is referred to a *false positive*. Correctly detected drifts are referred to as *true positive*. The number of experiments (each with one actual drift) are shown in Table H.3. As there are 3 experiments, there is a total of 3 concept drifts to be detected. In the table it is indicated how many *true* and *false positive* each method detected. As it can be seen, the developed method detected 3 concept drifts and only had 1 *false positive* detection. The best competitor in this regard, ADWIN, also detected 3 *true positives*, equals to the developed method. However, ADWIN has a very high *false positives* number of 6, compared with only 1 for the developed method. Thus ADWIN is triggering frequent and unnecessary adaptation to concept drift. Also, the remaining competitors found fewer *true positive* and a much higher number of *false positive* compared with the developed method.

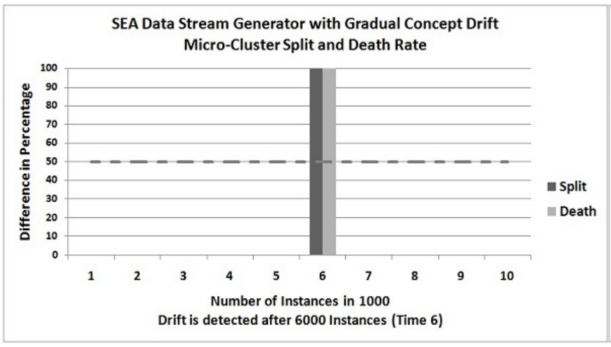
Table H.3: Summary of concept drift adaptation experiments highlighted in Table H.2.

Method	True Positives	False Positives
The Developed Method	**3**	**1**
CUSUM	1	1
DDM	2	2
EDDM	2	4
EWMA	2	18
ADWIN	3	6

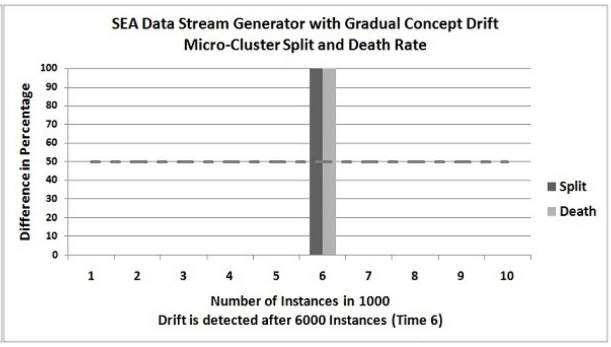
H.2.2 Real-Time Feature Tracking Method using Variance and IQR

This section now applies the in Sections H.2.1 discussed *Split* rate, *Death* rate and *LPF* in order to address tracking features. Furthermore this section also compares these results with the in Sections H.2.1 discussed new approach for building and adapting Micro-Clusters using *IQR* combined with feature *Velocity* instead of using *Variance* combined with feature *Velocity*. For the experiments the default parameters stated in Table H.1 of the method were used unless stated otherwise. Regarding the artificial datasets, no *Normalisation* was applied as the data generators already produced normalised data.

(a) Previous MC-NN (LPF and **Variance**)



(b) New MC-NN (LPF and **IQR**)



Features are ordered based on *Velocity* rate

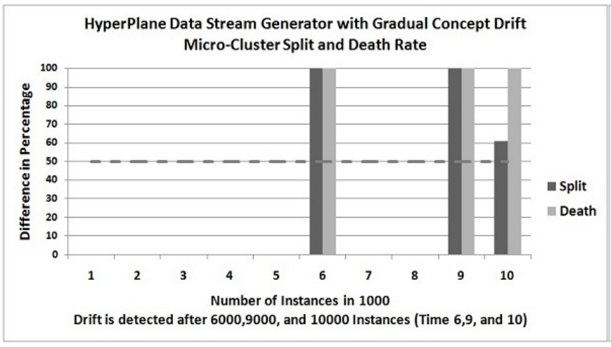
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 2 and 3)
Feature 1	0.00655	0	
Feature 2	0.00614	0	False Negative
Feature 3	0.00600	2	False Negative

Features are ordered based on *Velocity* rate

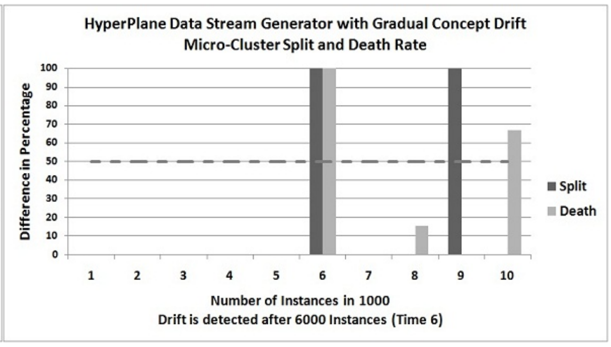
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 2 and 3)
Feature 3	**0.00483**	**9**	True Positive
Feature 1	0.00467	8	False Positive
Feature 2	0.00413	11	False Negative

Figure H.4: The results of *SEA* data stream generator using Micro-Clusters for tracking features.

(a) Previous MC-NN (LPF and **Variance**)



(b) New MC-NN (LPF and **IQR**)



Features are ordered based on *Velocity* rate

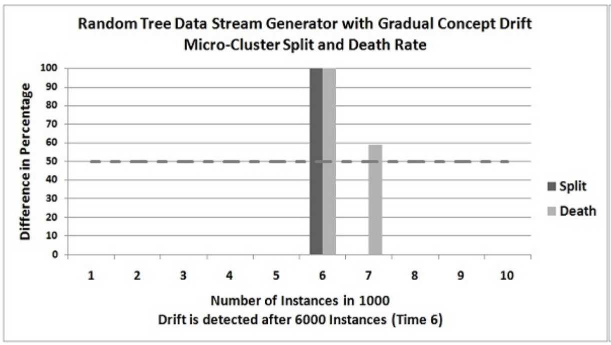
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 2 and 3)
Feature 3	**0.00111**	**3**	True Positive
Feature 2	0.00108	0	False Negative
Feature1	0.00098	0	

Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 2 and 3)
Feature 3	**0.00088**	**16**	True Positive
Feature 2	**0.00085**	**10**	True Positive
Feature 1	0.00072	4	

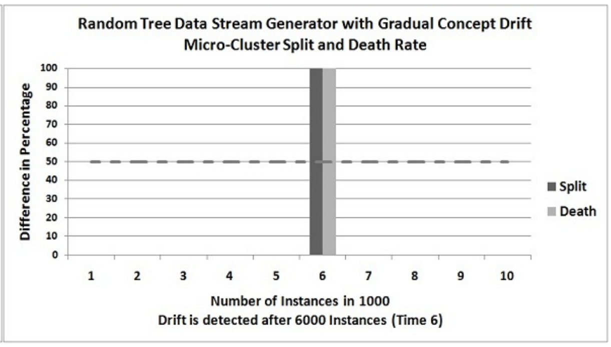
Figure H.5: The results of *HyperPlane* data stream generator using Micro-Clusters for tracking features.

(a) Previous MC-NN
(LPF and **Variance**)



Features are ordered based on *Velocity* rate			
Feature	**(Time 6) Velocity Rate (Descending Order)**	**(Time 6) History of Maximum Variance**	**Detection of Swapped Features (Feature 2 and 3)**
Feature 2	**0.00102**	**1**	True Positive
Feature 3	0.00094	0	False Negative
Feature 1	0.00080	0	

(b) New MC-NN
(LPF and **IQR**)



Features are ordered based on *Velocity* rate			
Feature	**(Time 6) Velocity Rate (Descending Order)**	**(Time 6) History of Maximum IQR**	**Detection of Swapped Features (Feature 2 and 3)**
Feature 1	0.00065	6	False Positive
Feature 2	**0.00057**	**4**	True Positive
Feature 3	0.00049	2	False Negative

Figure H.6: The results of *Random Tree* data stream generator using Micro-Clusters for tracking features.

Part (a) of Figures H.4 to H.6 shows the results for using MC-NN with *Variance* and history of maximum *Variance* for concept drift detection and feature tracking, and part (b) of the figures shows the corresponding results for using MC-NN with *IQR* and history of maximum *IQR*. As expected and similar to the results presented in Chapter 5 Section 5.4. Regarding concept drift detection, the same results as Section 5.4 have been achieved for both, using *Variance* and *IQR*. All concept drifts were detected correctly, however, the method based on *Variance* detected a false concept drift, see Figure H.5.

Table H.4 summarises the feature tracking results presented in Figures H.4 to H.6. It can be seen that the here presented method based on maximum *IQR* achieves a much higher *true positive* detection of features that changed. However, it also has a higher number of *false positive* detections compared with the method based on *Variance*. Now the *true positive* figures are based on the fact the features that are known to have changed as they have been swapped at the time of concept drift. However, considering the fact that each artificial stream generator's native method for inducing a concept drift has been used as well some of the *false positive* detections may very well be *true positives*. Even if they were not *true positives*, it would merely mean that they are flagged to the feature selection method to be reconsidered for inclusion or exclusion of the feature set to be considered for adaptation. Loosely speaking a high *true positive* detection numbers are more important than low *false positive* detection numbers.

Table H.4: Summary of the experimental results with artificial datasets generated. The results are reported for the Time 6 which is the time of swapped features.

Generator	True Positive (*Variance*)	True Positive (*IQR*)	False Positive (*Variance*)	False Positive (*IQR*)
SEA *with gradual concept drift*		1		1
HyperPlane *with gradual concept drift*	1	2		
Random Tree *with gradual concept drift*	1	1		1
Total:	2	4		2

Appendix I

Recurring Concept Drift Detection using Artificial Datasets

I.1 Experimental Setup of Artificial Datasets

Three datasets were generated using the MOA data stream generators (*SEA*, *HyperPlane*, and *Random Tree*), each comprising three features, two class labels and a concept drift. The concept drift was always induced halfway through the stream by swapping two features (i.e., induce a *recurring concept drift* as known ground truth), in order to show that the developed method is able to detect different types of concept drift. For example, features were swapped from 5200 instances to 5799 instances. The participation threshold Ω was set to 50 for each experiment, which yielded good results in most cases (Tennant et al., 2017). The error threshold Θ was set to the best performing one for each dataset. Table I.1 shows an overview of generated streams including the settings of the developed method and which features have been swapped. In the experiments, a window's size equals to 10% of the total number of instances. Hence, the expression **Time t** refers to a particular time window. I.e. according to the figures in this appendix time *T=1* refers to instances 1-1000, *T=2* to instances 1001-2000, etc.

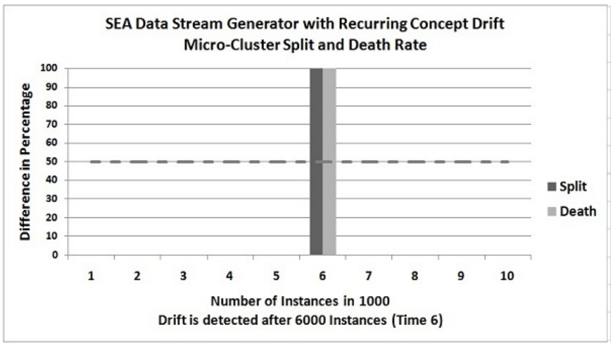
Table I.1: Setup of the artificial datasets. Drifts were generated by swapping features.

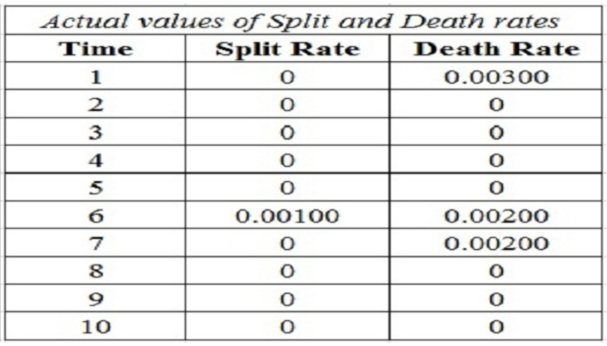
Dataset	Number of Instances Generated	Window Size	Θ	Index of Swapped Features	Start of Swapped Features
SEA	10,000	1000	16	2 with 3	6000
HyperPlane	10,000	1000	2	2 with 3	6000
RandomTree	10,000	1000	359	2 with 3	6000

I.2 Results

I.2.1 Real-Time Concept Detection Method

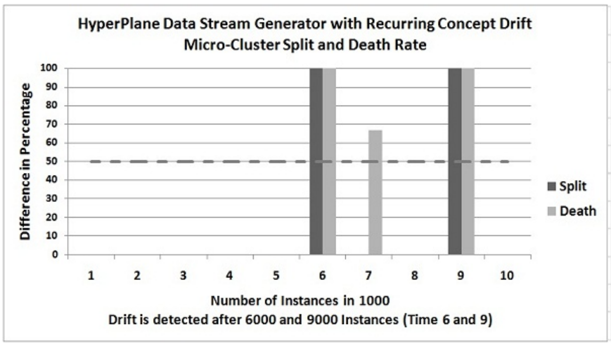
Micro-Clusters' *Split* and *Death* rates were used for detecting drifts. For the experiment the default parameters stated in Table I.1 of the method were used unless stated otherwise.

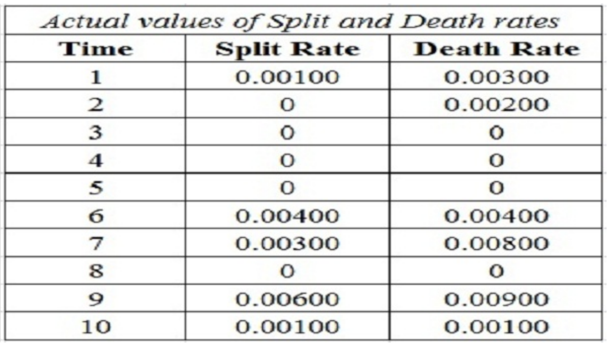




Actual values of Split and Death rates		
Time	**Split Rate**	**Death Rate**
1	0	0.00300
2	0	0
3	0	0
4	0	0
5	0	0
6	0.00100	0.00200
7	0	0.00200
8	0	0
9	0	0
10	0	0

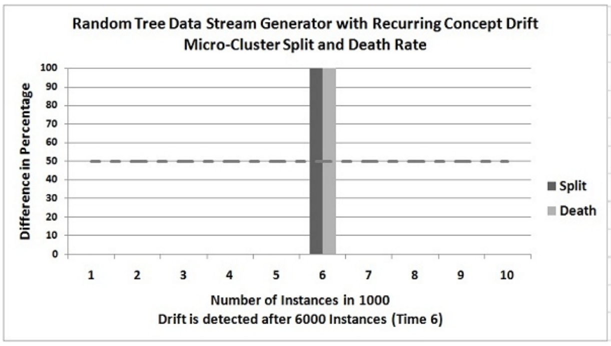
Figure I.1: The results of *SEA* data stream generator using Micro-Cluster *Percentage Difference* of *Split* and *Death* rates for drift detection.

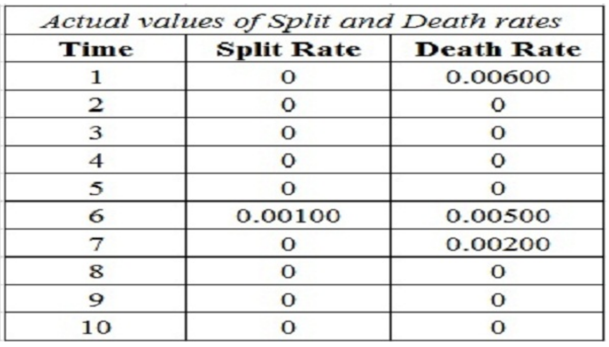




Actual values of Split and Death rates		
Time	**Split Rate**	**Death Rate**
1	0.00100	0.00300
2	0	0.00200
3	0	0
4	0	0
5	0	0
6	0.00400	0.00400
7	0.00300	0.00800
8	0	0
9	0.00600	0.00900
10	0.00100	0.00100

Figure I.2: The results of *HyperPlane* data stream generator using Micro-Cluster *Percentage Difference* of *Split* and *Death* rates for drift detection.





Actual values of Split and Death rates		
Time	**Split Rate**	**Death Rate**
1	0	0.00600
2	0	0
3	0	0
4	0	0
5	0	0
6	0.00100	0.00500
7	0	0.00200
8	0	0
9	0	0
10	0	0

Figure I.3: The results of *Random Tree* data stream generator using Micro-Cluster *Percentage Difference* of *Split* and *Death* rates for drift detection.

In Figures (I.1 to I.3), the focus is given more at the times when concept drift was introduced and features were swapped. During this time as higher *Split* and *Death* rates are expected as the set of Micro-Clusters adapts to the new concept the feature swap. In the figures, it can be seen that the *Split* and *Death* rates at the time of concept drift (after time 5) increase as expected for all artificial data streams. Next, the concept drift detection method was compared with existing state-of-the-art drift detection methods CUSUM, DDM, EDDM, EWMA, and ADWIN (see section 2 for more details about these methods) on the same artificial data streams. Table I.2 shows the time when each of the methods including the developed method, detected a drift and if it was detected on time. As it can be seen, the developed method always detected the drift at the correct time.

Table I.2: Adaptation to concept drift using the initially developed and other state-of-the-art methods.

Generator	Method	Number of Drift Detections	Times when Drift Detected	Drift Detected
SEA	**The Developed Method**	1	6	**Correctly**
	CUSUM	1	5	Incorrectly
	DDM	2	4 and 5	Incorrectly
	EDDM	3	2,3, and 5	Incorrectly
	EWMA	7	2 to 7 and 9	**Correctly**
	ADWIN	2	5 and 7	Incorrectly
HyperPlane	**The Developed Method**	2	6 and 9	**Correctly**
	CUSUM	-	-	-
	DDM	1	5	Incorrectly
	EDDM	2	6 and 8	**Correctly**
	EWMA	6	1,2, 4 to 6, and 8	**Correctly**
	ADWIN	1	2	Incorrectly
RandomTree	**The Developed Method**	1	6	**Correctly**
	CUSUM	1	5	Incorrectly
	DDM	1	5	Incorrectly
	EDDM	1	6	**Correctly**
	EWMA	8	1 to 5 and 7 to 9	Incorrectly
	ADWIN	3	4 and 6	**Correctly**

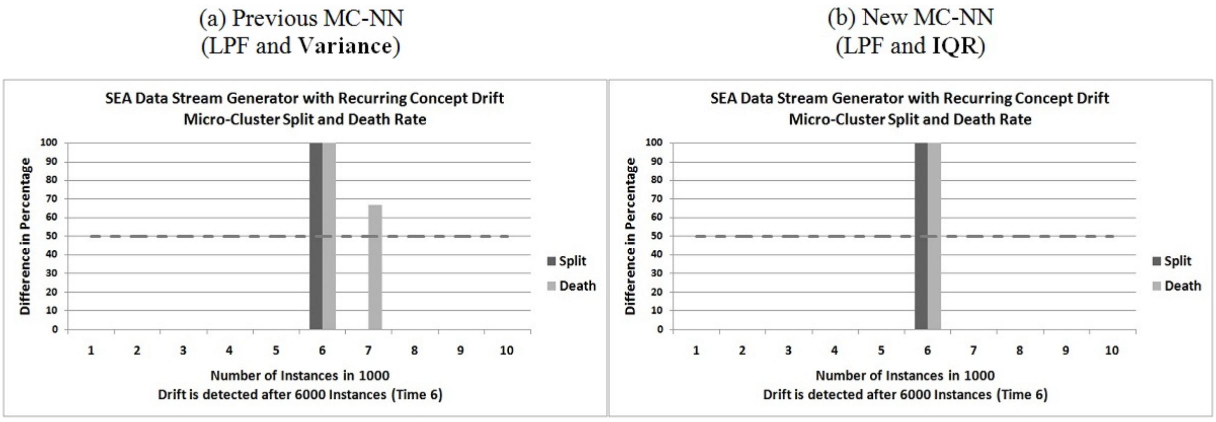
If a method detects a drift falsely, then this would require unnecessary adaptation by the classifier to a non-existing drift. This is referred to a *false positive*. Correctly detected drifts are referred to as *true positive*. The number of experiments (each with one actual drift) are shown in Table I.3. As there are 3 experiments, there is a total of 3 concept drifts to be detected. In the table it is indicated how many *true* and *false positive* each method detected. As it can be seen, the developed method detected 3 concept drifts and only had 1 *false positive* detection. The best competitor in this regard, EDDM and EWMA, detected 2 *true positives* only. However, EDDM and EWMA have a very high *false positives* number of 4 and 19, respectively compared with only 1 for the developed method. Thus EDDM and EWMA are triggering frequent and unnecessary adaptation to concept drift. Also, the remaining competitors found fewer *true positive* and a much higher number of *false positive* compared with the developed method.

Table I.3: Summary of concept drift adaptation experiments highlighted in Table I.2.

Method	True Positives	False Positives
The Developed Method	**3**	**1**
CUSUM		2
DDM		4
EDDM	2	4
EWMA	2	19
ADWIN	1	6

I.2.2 Real-Time Feature Tracking Method using Variance and IQR

This section now applies the in Sections I.2.1 discussed *Split* rate, *Death* rate and *LPF* in order to address tracking features. Furthermore this section also compares these results with the in Sections I.2.1 discussed new approach for building and adapting Micro-Clusters using *IQR* combined with feature *Velocity* instead of using *Variance* combined with feature *Velocity*. For the experiments the default parameters stated in Table I.1 of the method were used unless stated otherwise. Regarding the artificial datasets, no *Normalisation* was applied as the data generators already produced normalised data.



Features are ordered based on *Velocity* rate

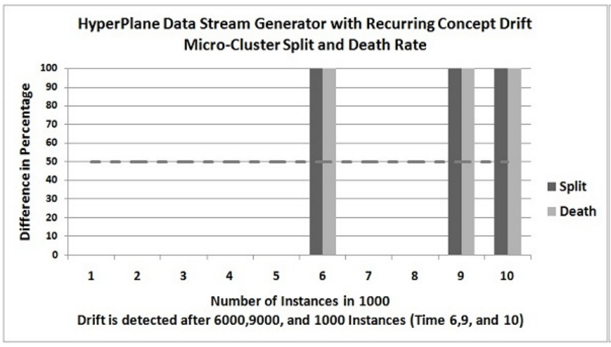
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 2 and 3)
Feature 3	**0.00255**	**1**	True Positive
Feature 2	0.00243	0	False Negative
Feature 1	0.00201	0	

Features are ordered based on *Velocity* rate

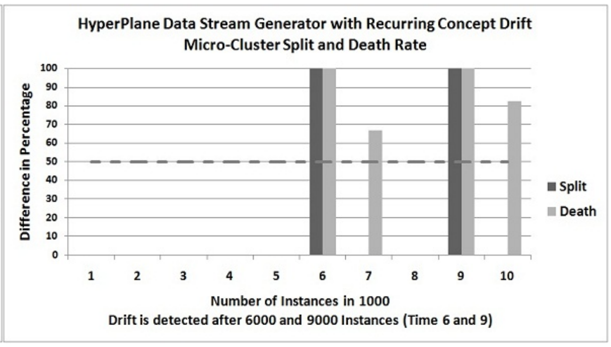
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 2 and 3)
Feature 3	**0.00097**	**2**	True Positive
Feature 1	0.00034	0	
Feature 2	0.00003	0	False Negative

Figure I.4: The results of *SEA* data stream generator using Micro-Clusters for tracking features.

(a) Previous MC-NN
(LPF and **Variance**)



(b) New MC-NN
(LPF and **IQR**)



Features are ordered based on *Velocity* rate

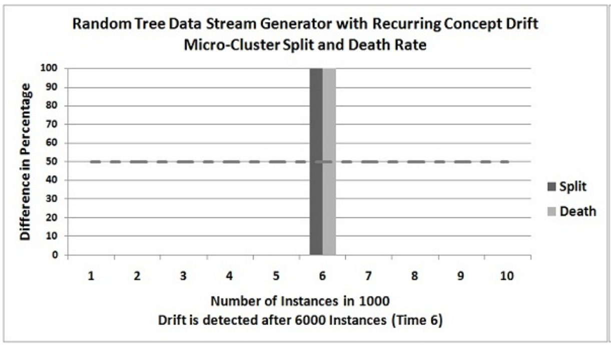
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 2 and 3)
Feature 3	**0.00115**	**2**	True Positive
Feature 2	**0.00115**	**1**	True Positive
Feature1	0.00103	0	

Features are ordered based on *Velocity* rate

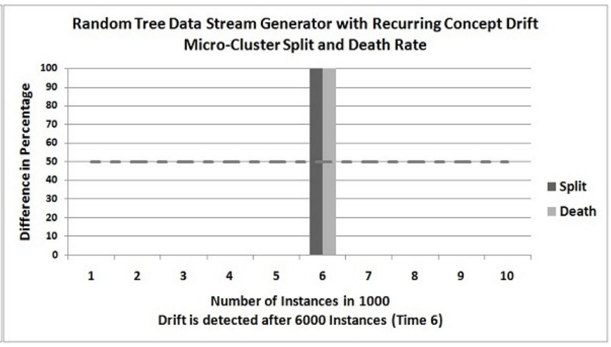
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 2 and 3)
Feature 3	**0.00023**	**4**	True Positive
Feature 2	**0.00022**	**1**	True Positive
Feature 1	0.00016	0	

Figure I.5: The results of *HyperPlane* data stream generator using Micro-Clusters for tracking features.

(a) Previous MC-NN
(LPF and **Variance**)



(b) New MC-NN
(LPF and **IQR**)



Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 2 and 3)
Feature 3	0.00245	0	False Negative
Feature 1	0.00242	0	
Feature 2	0.00240	1	False Negative

Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 2 and 3)
Feature 2	**0.00081**	**16**	True Positive
Feature 1	0.00076	31	False Positive
Feature 3	0.00072	5	False Negative

Figure I.6: The results of *Random Tree* data stream generator using Micro-Clusters for tracking features.

Part (a) of Figures I.4 to I.6 shows the results for using MC-NN with *Variance* and history of maximum *Variance* for concept drift detection and feature tracking, and part (b) of the figures shows the corresponding results for using MC-NN with *IQR* and history of maximum *IQR*. As expected and similar to the results presented in Chapter 5 Section 5.4. Regarding concept drift detection, the same results as Section 5.4 have been achieved for both, using *Variance* and *IQR*. All concept drifts were detected correctly, however, the method based on *Variance* detected a false concept drift, see Figure I.5.

Table I.4 summarises the feature tracking results presented in Figures I.4 to I.6. It can be seen that the here presented method based on maximum *IQR* achieves a much higher of *true positive* detection of features that changed. However, it also has a higher number of *false positive* detections compared with the method based on *Variance*. Now the *true positive* figures are based on the fact the features that are known to have changed as they have been swapped at the time of concept drift. However, considering the fact that each artificial stream generator's native method for inducing a concept drift has been used as well some of the *false positive* detections may very well be *true positives*. Even if they were not *true positives*, it would merely mean that they are flagged to the feature selection method to be reconsidered for inclusion or exclusion of the feature set to be considered for adaptation. Loosely speaking a high *true positive* detection numbers are more important than low *false positive* detection numbers.

Table I.4: Summary of the experimental results with artificial datasets generated. The results are reported for the Time 6 which is the time of swapped features.

Generator	True Positive (*Variance*)	True Positive (*IQR*)	False Positive (*Variance*)	False Positive (*IQR*)
SEA *with recurring concept drift*	1	1		
HyperPlane *with recurring concept drift*	2	2		
RandomTree *with recurring concept drift*		1		1
Total:	3	4		1