

DATA MINING AND RE-IDENTIFICATION: ANALYSIS OF DATABASE QUERY PATTERNS THAT POSE A THREAT TO ANONYMISED INFORMATION

by

Olabayo Ishola

Thesis submitted to the
De Montfort University Doctoral College
In fulfillment of the requirements
for Ph.D. Degree in
Cyber Security

Supervisors: Prof. Eerke Boiten and Prof. Aladdin Ayesh

Department of Cyber Technology Institute
De Montfort University, Leicester

July 2023

Declaration

I hereby declare that I am the sole author of this thesis. The work in this thesis is an original research work undertaken by me, submitted for the award of Doctor of Philosophy at the department of Cyber Technology Institute, De Montfort University, United Kingdom. There are no part of the work detailed in this thesis submitted for any other award, degree or qualifications in this or any other university or college of advanced education.

Olabayo Ishola
July 2023

Abstract

To maintain the globally connected civilization culture in place today, a number of sectors are built on the gathering and sharing of data. Personal and sensitive data are collected and shared about the individuals using the services offered by these sectors. Data controllers rely on the robustness of anonymisation measures to keep personal and sensitive attributes in the shared dataset privacy safe. Typically, the dataset is stripped of direct identifiers such as names and National Insurance (NI) numbers, such that individuals in the dataset are not uniquely identifiable. However, details in the dataset perceived by data controllers to have no negative data privacy impact can be used by attackers to perform a re-identification attack. Such an attack uses the details shared in the dataset in conjunction with a secondary data source to rebuild a personally identifiable profile for individual(s) in the supposedly anonymised shared dataset. There have been a few publicised cases of re-identification attacks, and with the information reported about these attacks, it is unknown what constitutes a re-identification attack from a technical perspective other than its outcome.

The work in this thesis explores real cases of successful re-identification attacks to analyse and build a technical profile of what re-identification entails. Using the Netflix Prize Data and the re-identification of Governor William Weld as case studies, synthetic datasets are created to represent the anonymised databases shared in each of these re-identification attack cases. An exploratory study to technically represent re-identification attacks as database queries in SQL is conducted. This involves the research performing re-identification attacks on the synthetic databases by executing a series of SQL queries.

With a hypothesis that there is enough similarity in the patterns of SQL database queries that lead to re-identification attacks on anonymised databases, this research employs data mining techniques and machine learning algorithms to train classifiers to recognise re-identification patterns in SQL queries. Four classification algorithms: Multilayer Perceptron (MLP), Naive Bayes (NB), K-Nearest Neighbors (KNN), and Logistic Regression (LR) are trained in this research to recognise and predict attempts of re-identification attacks. The results of the performance evaluation and unseen data testing indicate that the MLP, Multinomial Naive Bayes (MNB), and the LR classifiers are most effective at recognising patterns of re-identification attacks. During performance evaluation, the MLP classifier achieved an accuracy of 100%, the MNB achieved 79.3% and the LR achieved 100%. The unseen data testing shows that the MLP, MNB, and LR classifiers are able to predict new instances of re-identification attack attempts 79%, 71%, and 79% of the time respectively, indicating a good generalisation performance. To the best of this research's knowledge, the work in this thesis is the only effort to date to automate the recognition and prediction of re-identification attack attempts on anonymised databases. The novel system developed in this research can be implemented to improve the monitoring of anonymised databases in data sharing environments.

Keywords:

Data Anonymisation, Re-identification Attack, Data Privacy, Data Mining, Database, Netflix Prize Data, SQL Queries, Classification Algorithms

Acknowledgements

My deepest gratitude goes to my supervisor, *Professor Eerke Boiten*. Thank you for your support, insights and patience throughout my research journey. It's difficult to imagine myself doing this under someone else's supervision. I will also like to extend my appreciation to my second supervisor, *Professor Aladdin Ayesha*, for all your contributions towards the success of this work.

There are no words to qualify how thankful I am to my family for the love and support. *My parents* - thank you for all that you do for me. My brother *Jide* - thank you for letting me use your grammarly account. I hope this doesn't mean I have to stop, I'm too spoiled now. My sister *Lara* - thanks for all the prayers and moral support.

Thank you to my friends, *Kabiru* - for the conversations, motivations and always putting me up on all the latest *LaTeX* tricks. *Reem* - thank you for always helping me, can't wait for you to join the PhD club. *Niyi* - for always being up for trading PhD war stories. Thank you *Adham*, you are very much appreciated for always being available for consultancy. Thank you to members of the DMU Hackers Society (2020/2021) academic session that participated in the research data collection campaign.

Most importantly, thank you to the Almighty. For seeing me through life thus far, and for blessing me with all that it took to complete this milestone.

Dedication

Dedicated to my parents. Thank you.

Relevant Publication

1. Olabayo Ishola, Eerke Albert Boiten, Aladdin Ayesh, and Adham Albakri. Recognising re-identification attacks on databases, by interpreting them as SQL queries: a technical study. In *Privacy in Statistical Databases (PSD2020), Arezzo, Italy*, September 2020.

List of Acronyms

ADAM	Adaptive Moment Estimation
AdaGrad	Adaptive Gradient Algorithm
ADF	Anonymisation Decision Framework
AI	Artificial Intelligence
ANN	Artificial Neural Networks
AOL	America Online
API	Application Programming Interface
AUC	Area under the ROC Curve
BoW	Bag-of-Words
CSV	Comma-separated Values
CV	Cross-Validation
CNN	Convolutional Neural Network
DAE	Data Access Environment
DBMS	Database Management Systems
DoB	Date of Birth
DSA	Data Sharing Agreement
ERR	Error Rate
EU	European Union
FN	False Negatives
FNN	Feedforward Neural Network
FP	False Positives
FPR	False Positive Rate
GBR	Gradient Boosting Regression
GDPR	General Data Protection Regulation
GIC	Group Insurance Commission
GPDPR	General Practice Data for Planning and Research
GUI	Graphical User Interface
HES	Hospital Episode Statistics
LIBLINEAR	Large Linear Classification
IDS	Intrusion Detection System
IMdB	Internet Movie Database
KDD	Knowledge Discovery from Databases
KNN	K-Nearest Neighbors
LBFSG	Limited-memory Broyden-Fletcher-Goldfarb-Shanno
LR	Logistic Regression
MAE	Mean Absolute Error

MIT	Massachusetts Institute of Technology
MLP	Multilayer Perceptron
MSE	Mean Squared Error
NB	Naive Bayes
NEWTON-CG	Newton Conjugate Gradient Method
NHS	National Health Service
NI	National Insurance
NN	Neural Network
NPD	Netflix Prize Data
OVA	Open Virtual Appliance
PII	Personally Identifiable Information
PPDM	Privacy Preserving Data Mining
PPDP	Privacy Preserving Data Publishing
PVV	Positive Prediction Value
Q1	Quadrant 1
Q2	Quadrant 2
Q3	Quadrant 3
Q4	Quadrant 4
QI	Quasi Identifier
QoS	Quality of Service
RBAC	Role Based Access Control
ReLU	Rectified Linear Unit
RFR	Random Forest Regression
RMSProp	Root Mean Squared Propagation
RNN	Recurrent Neural Network
ROC	Receiver Operating Characteristics
RQ	Research Question
SAG	Stochastic Average Gradient descent
SGD	Stochastic Gradient Descent
Sklearn	Scikit-Learn
SQL	Structured Query Language
SSMS	SQL Server Management Studio
SVR	Support Vector Regression
TN	True Negative
TNR	True Negative Rate
TP	True Positives
TPR	True Positive Rate
TRE	Trusted Research Environment
VM	Virtual Machine
ZIP	Zone Improvement Plan

Table of Contents

Declaration	iii
Abstract	v
Acknowledgements	vii
List of Acronyms	xiii
List of Tables	xxi
List of Figures	xxiii
1 Introduction and Overview	1
1.1 Research Background	1
1.2 Research Gap	4
1.3 Research Questions	4
1.4 Research Hypothesis	4
1.5 Research Aim and Objectives	5
1.6 Research Methodology	5
1.7 Anticipated Outcomes	6
1.8 Limitations	6
1.9 Main Contributions	7
1.10 Thesis Outline	7
2 Data Anonymisation and Re-identification	9
2.1 Review of Related Work in Anonymisation and Re-identification	9
2.2 Data Anonymisation	15
2.3 Overview of Data Anonymisation Measures	16
2.3.1 k -Anonymity	16

2.3.2	<i>l</i> -Diversity	18
2.3.3	<i>t</i> -Closeness	20
2.3.4	<i>p</i> -Sensitive	20
2.3.5	Pseudonymisation	21
2.3.6	Perturbation	22
2.3.7	Differential Privacy	22
2.4	Data Re-identification	24
2.4.1	Objectives of Re-Identification Attacks	26
2.4.2	Overview of Data Re-identification Attempts	27
2.4.3	Rationales for Data Re-identification	29
2.5	Chapter Summary	30
3	Data Mining and Database Query Pattern Recognition	33
3.1	Data Mining (Machine Learning)	33
3.2	Overview of Data Mining Techniques	34
3.2.1	Predictive Data Mining	34
3.2.2	Descriptive Data Mining	35
3.3	Overview of Data Mining Tools and Evaluation	36
3.3.1	Data Mining Tool Evaluation Framework	36
3.3.2	Data Mining and Python Implementation	37
3.4	Review of work in Data Mining and Database Query Recognition	38
3.5	Data Mining and Re-identification Query Pattern Recognition	40
3.6	Supervised, Semi-supervised and Unsupervised Learning	41
3.7	Classification Problem in Re-identification Query Pattern Recognition	41
3.8	Application of Classification Algorithms in Re-identification Query Pattern Recognition	43
3.8.1	Neural Network	43
3.8.2	Naive Bayes	48
3.8.3	K-Nearest Neighbours	49
3.8.4	Logistic Regression	49
3.9	Classification Performance Evaluation Metrics	50
3.9.1	Bias and Variance	53
3.9.2	Cross-Validation	54
3.10	Chapter Summary	54

4	Data Re-identification Analysis and Case Study Selection	57
4.1	Exploratory Study Overview	57
4.2	Data Re-identification Case Studies	60
4.2.1	The AOL Data Release	60
4.2.2	Netflix Prize Dataset	60
4.2.3	The Re-identification of Governor William Weld	61
4.3	Recognising Re-identification Attacks as SQL Queries	61
4.3.1	Experimenting with Re-identification Attacks	62
4.3.2	Synthetic Data Creation	62
4.3.3	Re-identification Attack Re-creation	63
4.4	Re-identification Attack Experiment Analysis and Results	68
4.4.1	Experiment Analysis	68
4.4.2	Experiment Results	70
4.5	Experiment Strategy	72
4.6	SQL Query Data Collection	73
4.6.1	Data Collection Campaign	73
4.6.2	Data Collection Tools and Setup	74
4.7	Data Augmentation	77
4.8	Chapter Summary	79
5	Data Mining Models for Re-identification Query Pattern Recognition	81
5.1	Data Mining Experiment Overview	81
5.2	Data Preprocessing	82
5.2.1	SQL Query Data Preprocessing Libraries	82
5.2.2	Data Preparation and Labelling	83
5.2.3	SQL Query Data Augmentation	84
5.2.4	Feature Extraction	86
5.2.5	Data Splitting	88
5.3	Re-identification Query Pattern Recognition Using Binary Classification Algorithms with <i>Tokenizer</i> Feature Extraction	89
5.3.1	Multilayer Perceptron Classifier	90
5.3.2	Naive Bayes Classifier	92
5.3.3	K-Nearest Neighbours Classifier	93
5.3.4	Logistic Regression Classifier	94

5.4	Re-identification Query Pattern Recognition Using Binary Classification Algorithms with <i>TF-IDF</i> Feature Extraction	95
5.4.1	Multilayer Perceptron Classifier	96
5.4.2	Naive Bayes Classifier	97
5.4.3	K-Nearest Neighbours Classifier	97
5.4.4	Logistic Regression Classifier	97
5.5	Chapter Summary	98
6	Re-identification Query Pattern Recognition Models: Performance Evaluation and Analysis	99
6.1	Model Performance Evaluation	99
6.1.1	Evaluation of Re-identification Query classification with Multilayer Perceptron	100
6.1.2	Evaluation of Re-identification Query classification with Naive Bayes	101
6.1.3	Evaluation of Re-identification Query classification with K-Nearest Neighbors	109
6.1.4	Evaluation of Re-identification Query classification with Logistic Regression	111
6.2	Model Performance Evaluation Comparison	114
6.3	Understanding Misclassification in the Models	115
6.4	Unseen Data Test	117
6.4.1	Multilayer Perceptron	119
6.4.2	Multinomial Naive Bayes	121
6.4.3	K-Nearest Neighbors	123
6.4.4	Logistic Regression	125
6.5	Test Result Analysis	127
6.6	Chapter Summary	130
7	Conclusion and Recommendation	133
7.1	Introduction	133
7.2	Contributions	134
7.3	Future Work	137
7.4	Chapter Summary	137
	References	138
	APPENDICES	138

A Synthetic Dataset	151
A.1 Netflix Prize Data and GIC Databases	151
A.2 Data Collection Experiment Guide	152
B SQL Query Data	153
B.1 Normal Query Sample	153
B.2 Re-identification Attempt Query Sample	166
C Python Implementation	181
C.1 Data Preparation and Labelling	181
C.1.1 Random Word Masking Data Augmentation	182
C.1.2 Word Cloud for Random Word Masking Data	183
C.1.3 Random Word Insertion Data Augmentation	184
C.1.4 Word Cloud for Random Word Insertion Data	185
C.1.5 <i>Tokenizer</i> Feature Extraction	186
C.1.6 <i>TF-IDF</i> Feature Extraction	186
D Classification Experiment	189
D.1 Random Word Masking with <i>Tokenizer</i>	189
D.1.1 MLP Model Parameters	189
D.1.2 MLP Model Training	190
D.1.3 MLP classification report	190
D.1.4 Gaussian NB classification report	191
D.1.5 Bernoulli NB classification report	191
D.1.6 Categorical NB classification report	192
D.1.7 Baseline Multinomial NB classification report	192
D.1.8 Tuned Multinomial NB classification report	193
D.1.9 KNN with Euclidean Distance	193
D.1.10 KNN with Manhattan Distance	194
D.1.11 Baseline Logistic Regression	194
D.1.12 Tuned Logistic Regression	195
D.2 Random Token Insertion with <i>Tokenizer</i>	195
D.2.1 MLP Model Parameters	195
D.2.2 MLP Model Training	196
D.3 Random Word Masking with <i>TF-IDF</i>	202

D.3.1	MLP Model Parameters	202
D.3.2	MLP Model Training	203
D.3.3	MLP classification report	203
D.3.4	Gaussian NB classification report	204
D.3.5	Bernoulli NB classification report	204
D.3.6	Categorical NB classification report	205
D.3.7	Baseline Multinomial NB classification report	205
D.3.8	Tuned Multinomial NB classification report	206
D.3.9	KNN with Euclidean Distance	206
D.3.10	KNN with Manhattan Distance	207
D.3.11	Baseline Logistic Regression	207
D.3.12	Tuned Logistic Regression	208

List of Tables

2.1	A Comparison Review of Work in Anonymisation and Re-identification	15
2.2	Sample database I	17
2.3	β -anonymous table for sample database I	17
2.4	Sample database II	19
2.5	β -diverse table for sample database II	19
2.6	p -Sensitivity database table	21
3.1	A Comparison Review of Work in Database Query Recognition	40
3.2	Confusion Matrix for Binary Classification	51
6.1	Comparison of performance evaluation metrics in different MLP models	101
6.2	Comparison of different NB classification performance evaluation metrics using <i>Tokenizer</i>	107
6.3	Comparison of different NB classification performance evaluation metrics using <i>TF-IDF</i>	107
6.4	Comparison of baseline and tuned MNB classification performance evaluation met- rics (<i>Tokenizer</i>)	108
6.5	Comparison of baseline and tuned MNB classification performance evaluation met- rics (<i>TF-IDF</i>)	108
6.6	Comparison of classification performance evaluation metrics in KNN with Eu- clidean and Manhattan distance functions (<i>Tokenizer</i>)	111
6.7	Comparison of classification performance evaluation metrics in KNN with Eu- clidean and Manhattan distance functions (<i>TF-IDF</i>)	111
6.8	Comparison of baseline and tuned LR classification performance evaluation metrics (<i>Tokenizer</i>)	114
6.9	Comparison of baseline and tuned LR classification performance evaluation metrics (<i>TF-IDF</i>)	114
6.10	Comparison of performance evaluation metrics in re-identification attack attempt recognition models (<i>Tokenizer</i>)	115

6.11	Comparison of performance evaluation metrics in re-identification attack attempt recognition models (<i>TF-IDF</i>)	115
6.12	<i>Tokenizer</i> MLP classifier unseen data test result	120
6.13	<i>TF-IDF</i> MLP classifier unseen data test result	121
6.14	Unseen data test result for baseline and tuned MNB classifiers (<i>Tokenizer</i>)	122
6.15	Unseen data test result for baseline and tuned MNB classifiers (<i>TF-IDF</i>)	123
6.16	Unseen data test result for KNN classifiers with Euclidean and Manhattan distance functions (<i>Tokenizer</i>)	124
6.17	Unseen data test result for KNN classifiers with Euclidean and Manhattan distance functions (<i>TF-IDF</i>)	125
6.18	Unseen data test result for baseline and tuned LR classifiers (<i>Tokenizer</i>)	126
6.19	Unseen data test result for baseline and tuned LR classifiers (<i>TF-IDF</i>)	127
6.20	Comparison of unseen data test results for MLP, MNB, KNN and LR classifiers (<i>Tokenizer</i>)	129
6.21	Comparison of unseen data test results for MLP, MNB, KNN and LR classifiers (<i>TF-IDF</i>)	130

List of Figures

2.1	Joining two databases with redundant records for re-identification	25
3.1	Architecture of a Feedforward Neural Network	46
4.1	Re-identification Analysis and Case Study Selection Approach	59
4.2	Count function with <i>User ID</i> column as criteria	64
4.3	Count function with <i>Movies</i> column as criteria	64
4.4	Sorting with movie <i>Black Panther</i> as criteria	65
4.5	<i>Black Panther</i> entries on <i>IMDb</i>	66
4.6	Filtering <i>GIC</i> Dataset with the Governor’s DoB	67
4.7	Filtering <i>GIC</i> Dataset with the Governor’s DoB and Gender	68
4.8	Re-identification Likelihood Quadrant	69
4.9	Result for <i>User 7</i>	71
4.10	Result for <i>User 47</i>	71
4.11	Result for <i>User 43</i>	71
4.12	Result for Re-identification Attempt	72
4.13	Microsoft SSMS	75
4.14	Microsoft SQL Server Profiler	75
4.15	<i>General</i> Tab Configurations for the Trace Properties	76
4.16	<i>Events Selection</i> Tab Configurations for the Trace Properties	77
5.1	MLP classifier for re-identification attempt recognition (random word masking) . .	90
5.2	MLP classifier training accuracy and loss over 10 epochs (random word masking) .	91
5.3	MLP classifier training accuracy and loss over 10 epochs (random token insertion)	92
5.4	TF-IDF MLP classifier for re-identification attempt recognition	96
5.5	TF-IDF MLP classifier training accuracy and loss over 10 epochs	96
6.1	<i>Tokenizer</i> MLP classification performance evaluation confusion matrix and ROC curve	100

6.2	<i>TF-IDF</i> MLP classification performance evaluation confusion matrix and ROC curve	100
6.3	Classification performance evaluation confusion matrices for different NB classifiers with <i>Tokenizer</i>	102
6.4	Classification performance evaluation confusion matrices for different NB classifiers with <i>TF-IDF</i>	103
6.5	ROC curves for different NB classifiers using <i>Tokenizer</i>	105
6.6	ROC curves for different NB classifiers using <i>TF-IDF</i>	106
6.7	Tuned MNB classification performance evaluation confusion matrix and ROC curve (<i>Tokenizer</i>)	107
6.8	Tuned MNB classification performance evaluation confusion matrix and ROC curve (<i>TF-IDF</i>)	108
6.9	Classification performance evaluation confusion matrices for KNN with different distance metric (<i>Tokenizer</i>)	109
6.10	Classification performance evaluation confusion matrices for KNN with different distance metric (<i>TF-IDF</i>)	110
6.11	ROC curves for KNN with different distance metric (<i>Tokenizer</i>)	110
6.12	ROC curves for KNN with different distance metric (<i>TF-IDF</i>)	111
6.13	Classification performance evaluation confusion matrices for baseline and tuned Logistic Regression (<i>Tokenizer</i>)	112
6.14	Classification performance evaluation confusion matrices for baseline and tuned Logistic Regression (<i>TF-IDF</i>)	113
6.15	ROC curves for baseline and tuned LR classifiers (<i>Tokenizer</i>)	113
6.16	ROC curves for baseline and tuned LR classifiers (<i>TF-IDF</i>)	114
6.17	Word cloud depicting dataset words and their class memberships	116
6.18	Venn diagram showing the number of words in each class	117
7.1	Research Phases and Contributions	135
C.1	Word cloud for the normal class	183
C.2	Word cloud for the re-identification attack class	184
C.3	Word cloud for the normal class	185
C.4	Word cloud for the re-identification attack class	186
D.1	MLP model parameters	189
D.2	MLP model training	190
D.3	MLP classification report	190
D.4	Gaussian NB classification report	191
D.5	Bernoulli NB classification report	191

D.6	Categorical NB classification report	192
D.7	Baseline Multinomial NB classification report	192
D.8	Tuned Multinomial NB classification report	193
D.9	KNN with Euclidean Distance classification report	193
D.10	KNN with Manhattan Distance classification report	194
D.11	Baseline Logistic Regression classification report	194
D.12	Tuned Logistic Regression classification report	195
D.13	MLP model parameters	195
D.14	MLP model training	196
D.15	MLP classification performance evaluation confusion matrix and ROC curve	196
D.16	Classification performance evaluation confusion matrices for different NB classifiers	197
D.17	ROC curves for different NB classifiers	198
D.18	Tuned MNB classification performance evaluation confusion matrix and ROC curve	199
D.19	Classification performance evaluation confusion matrices for KNN with different distance metric	199
D.20	ROC curves for KNN with different distance metric	200
D.21	Classification performance evaluation confusion matrices for baseline and tuned Logistic Regression	200
D.22	ROC curves for baseline and tuned LR classifiers	201
D.23	MLP model parameters	202
D.24	MLP model training	203
D.25	MLP classification report	203
D.26	Gaussian NB classification report	204
D.27	Bernoulli NB classification report	204
D.28	Categorical NB classification report	205
D.29	Baseline Multinomial NB classification report	205
D.30	Tuned Multinomial NB classification report	206
D.31	KNN with Euclidean Distance classification report	206
D.32	KNN with Manhattan Distance classification report	207
D.33	Baseline Logistic Regression classification report	207
D.34	Tuned Logistic Regression classification report	208

Chapter 1

Introduction and Overview

The increase in the amount of data available in the modern society, and the need to process these data for information and knowledge maturation necessitates data sharing. However, data sharing is not without risks, as personal and sensitive information about individuals in a shared dataset is susceptible to being linked to the data subjects, even if the dataset is anonymised when shared. This chapter addresses data privacy issues around data anonymisation and how re-identification is a threat. Section 1.1 presents detailed background information and the problem domain for this research. The knowledge gap and research questions posed by this work are in section 1.2 and 1.3 respectively. Section 1.6 summarises the overall methodology adopted in this research and the main contributions as a result of the work done in this research are highlighted in section 1.9. The outline for the thesis is presented in section 1.10.

1.1 Research Background

As information technology becomes more prominent in every sector of government administration, private organisation and regular human interaction, the amount of existing personal data about each individual has immensely multiplied over the years. Different arms of the government and private organisations hold information about a large number of individuals for different aims and purposes. Organisations such as hospitals hold medical records for treatment and research purposes, while government bodies such as different arms of the military, hold information such as mission details of military personnel. Governments and organisations need to sometimes release datasets, for research, statistical analysis to explore new inferences, public access and knowledge maturation. This type of dataset includes Hospital Episode Statistics (HES); a database with details about admitted patients and outpatients appointments at NHS hospitals in England [38]. This is a database with personal and sensitive information about individuals. Research [148, 107, 109] has proven that there is a risk of having sensitive personal information (for example, a medical condition of a specific person) about an individual revealed in the process or as a result of data sharing [57].

Often, data holders release and share their databases with third parties, including marketing organisations and commercial data brokers [96]. However, there are data protection laws such as the EU's General Data Protection Regulation (GDPR) that limit the exploitation or sale of personal (sensitive) information when data is shared or sold. Data protection laws intend to protect personal data, they control and mould the actions of institutions and government

bodies regarding personal data in the modern era of information. Private institutions and the government alike have continually proved that without strict data protection laws overseeing the protection of personal data, they (organisations and government) intend to collect, mine, store and share databases they hold without restraint, notification or accountability [1]. Although data distribution provides useful information to researchers and data analysts, the need to share datasets has brought about privacy risks to individuals whose data is being distributed [154].

The need to share data without revealing personally identifiable details brought about the concept of *anonymisation* in information sharing. A dataset is considered to be anonymised if it has been perturbed or generalised, such that it is impossible to attribute sensitive information to a specifically identified individual. In theory, once a database has been “anonymised”, it cannot be used to identify any specific individual or be used to deduce any of their sensitive information. Therefore, data protection regulations are less strict about the sharing of anonymised datasets for research, market analysis and other third-party purposes. Data holders are more inclined to share their datasets with other parties because of their trust in anonymisation techniques as a data privacy measure. Interestingly, the rapid increase in the amount of publicly available information over the internet and other publicly accessible databases, coupled with advanced computer hardware and software with remarkable data processing ability has made it possible to analyse and “re-identify” identifiable information from datasets that might have been considered anonymised [96].

Anonymised data are typically re-identified by combining two or more databases, in search of fragments of information that may suggest that the information from these databases are about the same individual, or by having information about the same entity stored multiple times (data redundancy). When the process of re-identifying an individual from an anonymised dataset is performed successfully, it has dire data privacy protection repercussions, as it may violate the conditions under which the information was divulged by the data subject, or collected and shared by the data holder. This is referred to as a *Re-identification Attack*.

As mentioned in [110], a foundational belief about data privacy states “Data can either be useful or perfectly anonymous but never both”. “Useful” in this instance would mean that the dataset is minimally anonymised or not at all, making it suitable for third-party analysis, or the dataset could be “perfectly anonymised”, such that it is difficult or impossible to make meaningful inferences when analysed. This project is interested in rather effectively anonymised datasets and the extra lengths attackers would go to re-identify personal details in the datasets. It is imperative to strike a compromising balance between an un-anonymised, useful dataset and a strictly anonymised unusable one. An exploration into the anonymity/usability balance brings forth the idea of minimally anonymised but monitored datasets, which allow third party analysis on the data. The monitoring process is automated and it is geared towards proactive detection of possible re-identification attacks. This is the focal point of this research work.

Relating *Personal Identifiable Information* (PII), *Personal Data* or *Sensitive Data*, such as name or NI (National Insurance) number to a specific individual are the usual goal an attacker aims for when performing re-identification attacks. Even though PII is the usual target, “other information” about the data subjects which are traditionally presumed not to be personally identifiable can also be used to re-identify subjects from anonymised datasets. Attributes that are not considered personal or sensitive (*other information*), combined to achieve re-identification are what Latanya Sweeney defined as *quasi-identifier* [130, 110, 109]. Quasi-identifiers are sets of information that do not uniquely identify an individual, but can be combined with other quasi-identifiers or external data sets to generate a unique identifier [57, 50]. For instance, a

study by Sweeney found that 87% of the United States population is uniquely identifiable with the combination of their ZIP code, date of birth (DoB) and sex [149]. Similarly, date of birth and ZIP code makes 97% of the population of Cambridge, Massachusetts identifiable [109]. The work in these studies emphasises that data controllers and data privacy professionals do not classify *quasi-identifiers* as PII, even though quasi-identifiers are essential to the execution of re-identification attacks.

Information to be released by an organisation is released as microdata: these are units of data that aggregate statistics are derived from, consisting of records including information on individuals or business entities [28]. Identifying characteristics such as names or national insurance numbers are usually not released, however, the microdata contains quasi-identifiers, which poses a threat to the privacy of the information. Attackers can use quasi-identifiers to reveal the individual the information is about.

There have been attempts by UK government officials to gather and share health data, with the belief and claim that anonymisation is enough to keep the data privacy safe. An instance of this, is the General Practice Data for Planning and Research (GPDPR) scheme by the UK government. The scheme had planned to upload millions of NHS users' data to a central database. Their privacy justification for this is that the data will be *pseudonymised*. The GPDPR plan was postponed after objections were made by various medical bodies [17]. The high sensitivity of health data and the need for it to be shared safely for useful causes (e.g., health research) has motivated innovations towards the privacy safe or usability dilemma that has plagued data sharing.

Trusted Research Environment (TRE) service for England is a service by NHS Digital that provides trusted researchers secure access to health datasets upon approval. Specific Data Sharing Agreement (DSA) is established between the researchers and NHS Digital, then access is provided to the approved dataset within the TRE service that enables the researchers to collaborate, link the data and share results within the same research project. All data available on the platform is anonymised, with personal details such as names or NHS Numbers removed. To make use of the data, the TRE service provides researchers access to the Data Access Environment (DAE). An environment incorporated with powerful data analysis and query tools including Databricks (a collaborative tool used for data analysis with SQL and Python languages support). The overall intended output generated by the researchers is reviewed for compliance [37].

This research is aiming to develop a system that can be deployable in the creation or maintenance of privacy-safe data sharing and processing conditions of a TRE. The anticipated product of this work can be applicable as a feature in a platform such as the NHS Digital TRE service. It can be used to foster a higher level of control and accountability in how anonymised datasets are being accessed. The TRE service DAE supports the use of SQL queries, therefore the system developed by this research work can be utilised on the TRE service for England platform to inspect SQL queries executed by the researchers against datasets on the DAE for possible re-identification attacks. The focus of this work is to try and understand what the technical features of data re-identification attacks are, in the SQL queries executed on databases. This knowledge will be used to gather relevant data, use these data to train a system that will be capable of recognising the signatures of re-identification attack attempts on an anonymised database. This system will be trained using data mining techniques.

1.2 Research Gap

In the modern era of big data, society can make many advances through large scale analysis of databases and combinations of databases. However, these may contain personal information and as a consequence, there are risks of unethical or illegal use of such information. Research and publications [110, 148, 149, 88] have addressed the inefficiency of anonymisation techniques to completely preserve the privacy of anonymised databases, particularly in the context of data releases. These publications on the link between anonymisation and re-identification tackled the topic from the perspective of warning data holders about placing too much faith in the power of anonymisation techniques: by emphasising on how relatively easy it is to re-identify the anonymised details in a database [110, 107].

The bulk of the available measures against re-identification attacks within the data privacy field includes suggestions for better anonymisation models, human discretion or legal actions against performing re-identification. There is no evidence of research work conducted into the exploration of database search query analysis or the application of automated systems to identify re-identification attack attempts. By generating a model of what constitutes a re-identification attack in a technological context, this research develops a system that automatically and intelligently detects malicious database queries that may be attempting to re-identify redacted details of an anonymised database. The system is developed by adopting principles of data mining and implementing machine learning algorithms.

1.3 Research Questions

In order to fulfil the purpose of this project, answering the questions stated below is an essential part of the overall process. The research questions addressed in this thesis are:

1. How can the anonymisation measures that have been applied to a dataset be compromised through database queries?
2. What does a re-identification attack look like from a technical perspective?
3. Is it possible to automatically recognise re-identification attacks through studying patterns in SQL queries executed on anonymised databases?
4. How can data mining techniques be applicable in database queries, to train an intelligent system that can monitor and recognise re-identification patterns in these queries?

1.4 Research Hypothesis

The hypothesis of this project is that re-identification attacks have enough in common with each other that they can be systematically discovered when they are in progress. For example, as particular sequences of database requests. Initially, the work in this research establishes what a re-identification attack on a anonymised database is from a technical perspective. Then the research implements machine learning algorithms to learn and recognise patterns in database queries that insinuate re-identification attacks.

1.5 Research Aim and Objectives

This research focuses on the concept of data anonymisation and re-identification, investigating queries on anonymised databases for patterns that may pose a threat to data privacy. Using machine learning models, the research will explore how systems can intelligently learn and recognise database query patterns that attackers may employ to exploit the vulnerabilities in the anonymisation measures used for storing data. To achieve the aim of this research, the following are the key objectives that will be pursued:

1. To engage in a comprehensive review of the existing relevant literature on data privacy as it pertains to anonymisation, re-identification, and explore how machine learning can be applied to achieving the aim of the research.
2. To establish that re-identification attacks are achievable by executing a series of database queries on anonymised databases, and develop a model of what possible re-identification queries look like (by experimenting with re-identification attacks on anonymised databases).
3. To develop a prototype system that monitors and recognises when queries that fit into the re-identification model are noticed in a database system.
4. In the event that re-identification does not constitute a recognisable pattern, the research work aims to present its justification and conduct further research to explore other dimensions of the relationship between anonymisation, re-identification and data mining.
5. To produce a PhD thesis that presents the methods employed, experiments conducted and results generated in the process of achieving the aim of this research.

1.6 Research Methodology

With the intention of this research to achieve the set out aim and objectives, the research analysis is presented using a qualitative approach. This research explores the intricacies of existing concepts, case studies and publications relating to the research area. The work also consists of literature review into all the relevant concepts of data anonymisation, re-identification and data mining, focusing on how these elements can be used towards realizing the research aim and objectives. In the course of attempting to fulfil the requirements of the objectives for this research, access to an anonymised database is essential. However, efforts to acquire an anonymised health database have been unsuccessful, reaching out to relevant officials of the UK NHS (National Health Service) and discussing the details of what the project is aiming to achieve. The discussion emphasised the obvious benefits that the results of this work will offer to an organisation such as the NHS (significantly improve measures to keep anonymised health data private and anonymous). With the research's inability to acquire real anonymised datasets, synthetic datasets are therefore generated (modelled on real-life re-identification scenarios) and anonymised for the purpose of this project.

Analysis of the generated synthetic data was conducted, which involved applying anonymisation measures to the dataset. This is necessary to keep the generated dataset in the state that it would have been when shared with the public. The stories in the published re-identification attack scenarios used were closely studied and this informed the choice of anonymisation criteria applied

to the synthetic dataset. An experiment whereby re-identification attacks are attempted against the anonymised dataset was conducted. The experiment involved attempting re-identification attacks using Structured Query Language (SQL) queries to interact with the database. This was done to establish that re-identification is achievable by executing a series of SQL queries against the target dataset. With this, a baseline for what constitutes a re-identification attack (from a technical perspective) was established. The experiment and its results have been published in a peer-reviewed conference paper [76], the work published in the paper is mostly represented in chapter 4 of this thesis. The author of this thesis is the primary and first-named author of the paper.

A data collection campaign to gather SQL query dataset was conducted, this required participants in the data collection activity to attempt re-identification attacks against the synthetic anonymised databases created for this research. SQL query logs for the queries executed by participants are recorded. Data Mining techniques were then introduced to train and develop a system that can systematically identify user queries that may be attempting a re-identification attack. The system was trained to automatically recognise (using its knowledge of re-identification attack attempt query patterns) attempts at re-identification. Different machine learning models were explored and trained, and the most suitable for the purpose of achieving the aim and objectives of this research work are selected and implemented.

1.7 Anticipated Outcomes

Success in this research will contribute to creating or maintaining a practical data sharing scenario that transcends the typical unsupervised de-identified (anonymised) database that solely relies on the strength of its anonymisation technique for privacy protection or a heavily supervised database with no anonymisation. A definite disadvantage of the former is the fact that anonymisation techniques are imperfect, and the latter will require intensive manpower to manage. A positive result in this research will create a compromise by making the administration of privacy-safe, moderately supervised, anonymised database sharing possible. The reduced supervision will be complemented by an automated system to be developed by this research work. The system will automate the way instances of possible re-identification attacks are detected on anonymised databases.

On the other hand, a negative outcome is a possibility in this research work. However, this need not be deemed as a failure, provided that a negative outcome will be systematically analysed to understand all the nuances involved in attempting to automatically recognise and subvert re-identification attacks through studying user query patterns.

1.8 Limitations

In the course of attempting to fulfil the objectives of this research work, there were challenges encountered. This has led to some perceived limitations the research is subjected to, which include:

1. To practically determine how data re-identification works, and also develop a template of search queries that may be malicious to anonymised datasets, real-life anonymised datasets

will productively contribute to the research experiment. However, efforts to acquire real, anonymised datasets have been to no avail. This research has made an attempt to acquire real anonymised datasets from the NHS, but the attempt was unsuccessful. Also, publications [107, 109, 3, 33] that reported cases of successful re-identification in their work did not share the datasets used in their re-identification experiments.

2. From the work done by this research into studying different existing data re-identification scenarios, evidence shows that some re-identification attacks may not be a product of technical database search queries and may require further work.
3. There is also a limitation in the amount of data available for usage in the process of this research, especially as it pertains to the number of re-identification case studies to analyse and explore during experimentation with re-identification attacks.

1.9 Main Contributions

Typically, data re-identification attacks could be of different forms. It could be as a result of social engineering or investigative techniques; to gather additional information that complements the redacted details of an anonymised dataset (this can be inferred from the details of famous re-identification attacks). Therefore, there has not been research work done to explore and ascertain what constitutes a re-identification attack from a technical perspective. With the rapid advancement of modern information technology solutions, relational Database Management Systems (DBMS) used for storing anonymised datasets are accessed and interacted with using SQL. This research is providing a technical representation of re-identification attack attempts in the form of SQL database queries. This contributes a unique insight into data privacy and data sharing, specifically to the study of data re-identification attacks on anonymised databases.

The results of unique experiments in this research produced a system that automatically recognises attempts of data re-identification attacks on anonymised databases. The system studies the patterns in the SQL queries that users are executing on the database. As a result of SQL query analysis, implementation of data mining techniques and machine learning algorithms, the resulting system will automate the vetting of database queries for re-identification attempts. This will be directly useful in real-world applications where data privacy is of high priority. NHS Digital's TRE service for England is an example of a platform where the system produced by this research can find a real-world application.

1.10 Thesis Outline

The contents in chapter 1 covers the conceptual overview of anonymisation and how data handlers place an exaggerated trust in data privacy that anonymisation measures offer. Re-identification is also introduced in this chapter, as a formidable threat and form of attack against anonymisation. The aim for this research is presented in this chapter, which is to develop a system that can recognise database query pattern that may be attempting re-identification attack and pose a threat on data privacy. This aim is explored by answering the research questions set out in 1.3 and fulfil the research objective breakdown in section 1.5. The aim, objectives and research questions are addressed in the different sections presented in this chapter, covering the contributions to knowledge offered by this research work.

Chapter 2 explores and reviews different published literature and research work that relates to data anonymisation and re-identification. The reviewed work approached these concepts from different perspectives and proposed a different solution to the data privacy threat that re-identification of anonymised dataset poses, this is presented in section 2.1 of this thesis. The exploitation of the de-identified (anonymised) data by attackers to achieve re-identification is also presented in this chapter. The concepts of anonymisation, and re-identification and how they relate to each other are also explored.

Chapter 3 starts with a general overview of data mining and the application of machine learning algorithms in solving different data mining problems. The categorisation of data mining tasks is explored and an overview of data tools evaluation criteria is presented. Chapter 3 presents this research's plan to implement Python data mining libraries for the classification problem in this work. This chapter further explores different algorithms used in the training of classifiers to recognise SQL query patterns that may be attempting a re-identification attack on an anonymised database. A review of existing related to the application of machine learning algorithms in database query recognition is also presented.

The explorations in Chapter 4 of this thesis introduce different data re-identification case studies that are analysed by this research and how these case studies are used to verify that re-identification attacks can be recognised as a series of SQL database queries. The creation of the synthetic databases used for experimentation at this stage is detailed. Re-identification attack scenarios are technically recreated during this phase of the research. The strategy employed for the recreation of different re-identification attacks and analysis of the results from the experiment is presented. Using the synthetic database created, a data collection campaign is used to gather a research experiment dataset to be used in the data mining application.

In Chapter 5, the approach employed to process the research dataset is presented. Data processing libraries in Python programming language are used for data preparation, labelling, augmentation, tokenization and feature extraction. The Python algorithms used for this data preprocessing task are presented in this chapter. The methodology and strategy employed in training the implemented classification algorithms are also presented in this chapter.

Chapter 6 details the performance evaluation for the classification models implemented for the re-identification query recognition in this research. Different performance evaluation metrics for classification algorithms such as accuracy, precision, recall and f-score are used to evaluate the models. Unseen data samples are used to test the ability of the classification models to recognise new instances of re-identification attack attempts and the analysis of the results is presented.

Chapter 7 presents conclusive discussions around the research conducted in this thesis, highlighting the achievements and academic contributions. Different related areas that can be explored for further research endeavours are also highlighted.

Chapter 2

Data Anonymisation and Re-identification

Data re-identification is a form of data privacy attack against anonymisation in data collection, data storage and most especially, data sharing. Prior to the increased possibility and awareness of re-identification attacks, data controllers could release anonymised datasets publicly. Data analysts from different fields of research could use the dataset about unidentified and unidentifiable individuals for analyses that are beneficial to their respective sectors. Many sectors rely on the use of shared datasets as a means of gaining reliable insights which decisions will be based upon. The health sector, for example, shares datasets to further improve the quality of research conducted in the sector. An instance of this is the HES dataset: a dataset containing medical details about patient diagnoses; personal details such as age group, gender and ethnicity; administrative details; and geographical details such as a patient's area of residence or treatment. The HES dataset is processed by NHS Digital for both clinical purposes (relating to patient care) and non-clinical purposes (secondary uses) that include analysing and monitoring trends and patterns in NHS hospital activities, planning health services, and research [38]. Different publications have addressed the relationship between data anonymisation and re-identification. An analytical review of the published work is presented in 2.1. This chapter explores the different elements surrounding the concepts of anonymisation and the measures used in its implementation as presented in sections 2.2 and 2.3 respectively. Concepts in data re-identification are covered in section 2.4.

2.1 Review of Related Work in Anonymisation and Re-identification

Samarati and Sweeney in [130] emphasized the increased appetite for information in modern society due to access to the internet and inexpensive computing. Often, the information is subjected to being shared, exchanged and sold. The paper emphasised the ramifications of improper handling of information (shared, exchanged or sold) can be damaging to individuals and organisations, particularly when the information is sensitive or personal. This publication criticised a prevalent belief held by data holders, both private and government agencies, that removing all direct identifiers from a set of data provides a scenario to share such a dataset without compromising the anonymity of individuals in the dataset. Samarati and Sweeney explicitly

stated that removing direct identifiers from datasets does not equate to guaranteed anonymity of subjects of the dataset. This statement was supported by Sweeney’s work in [148], proving that with the combination of basic demographic attributes such as ZIP code, DoB, gender, ethnicity and marital status, individuals in an anonymised dataset can be uniquely identifiable.

[130] explored the application of *generalisation* and *suppression* towards a solution against the re-identification. The paper introduces the definition of *quasi-identifiers*, a term coined by [29], as attributes that can be exploited by malicious users to attempt a re-identification attack. *k*-anonymity is also introduced in this work as a technique to characterise the degree of privacy protection of the dataset. Ensuring *k*-anonymity in a dataset after release with the application of generalisation and suppression is presented in this work. *k*-anonymity requires the specification of a minimum number *k* of duplicates of each record (row) with respect to the attributes (columns) of the quasi-identifiers. A detailed discussion on *k*-anonymity is presented in 2.3.1. The paper concluded by acknowledging that the method presented was a first step towards controlling data sharing, admitting that definitions of quasi-identifiers and finding an appropriate size of *k* are issues to be addressed. The work postulates that a generalised dataset is of the best quality when the subjects’ most important attributes are not a part of any quasi-identifier.

In 2002, Latanya Sweeney in a journal article about *k*-anonymity as a model for protecting privacy [149], explores the complexity and contradiction in the need for personal data holders such as banks and hospitals to release and share this information while simultaneously ensuring that the data subjects cannot be re-identified, at the same time sharing this data in a format that is practically useful. This work provided a privacy protecting framework known as *k*-anonymity; a data anonymisation technique used for modifying databases, such that no user of the database can deduce a probable link between entry details in the database and the individuals the details describes. Deployment policies for the framework were also presented in this paper. The paper further explored the re-identification attacks that may be accomplished if the policies for the framework are not adhered to. In the background work for the paper, Sweeney briefly touched on how multiple queries could leak inference to sensitive information in the database [149].

Paul Ohm [110], in his article, “Broken Promise of Privacy: Responding to the Surprising Failure of Anonymisation”, emphasised the disappointments of anonymisation techniques. The article highlighted the faith in anonymisation to provide data privacy-protecting functionalities, and allow databases to be freely shared to serve a variety of purposes. However, computer scientists have established that individuals in anonymised or de-identified databases can be re-identified or de-anonymised with surprising ease. Ohm pointed out that the magnitude of assurance placed in anonymisation was a mistake and that nearly every data privacy law, as well as regulators, paid little attention to the shortcomings of anonymisation, such as the ease with which it can be re-identified. The article provides tools to respond to these shortcomings. One of the clearest early expositions of the limitations of anonymisation as a privacy preservation measure discussed in this article is that, even under the supervision of sophisticated data handlers, different scenarios have discredited the idea that simply removing PII from datasets can protect privacy. Ohm emphasised that prior to studies such as work in [130, 148, 107], data holders did not consider features such as DoB, ZIP Code or movie ratings as PII - features that are now proven to be usable in executing successful re-identification attacks. However, Ohm stated that even after these studies, organisations and data holders have proceeded with releasing these types of features connected to sensitive data in supposedly anonymised databases.

The paper goes as far as pointing out that data technologists and even non-technical users of databases should stop the usage of the term “anonymise” or “de-identified”, as it insinuates

success in achieving anonymity of data subjects. Rather he proposed the term “scrub” as a fitting alternative because it only suggests attempt and effort, not success. Latanya Sweeney, in [148] similarly criticised the term “anonymise” in favour of “de-identified”. Both authors are alluding to how imperfect anonymisation techniques are in achieving privacy protection.

Ohm [110] also presents some analysis of how these limitations may be tackled. With the traditional practice being that the data holder performs anonymisation before the release of the dataset (referred to as “release-and-forget”), an improved technique was suggested. This entails an interactive data release relationship, as the data holder maintains constant participation after the data is shared. However, this is also characterised in the paper as not being a perfect approach, because of the increase in the cost requirement. Another argument towards a solution was made in this work and it pushes for a ban on re-identification, that data privacy law should simply state that any anonymised data is off limit and must not be re-identified. However, this is not a realistic measure because re-identification can happen in settings where they cannot be detected and without the knowledge of the organisation that released or shared the dataset. Achieving a successful re-identification ban is not feasible, because such a ban will be impossible to enforce. Legal measures such as the UK Data Protection Act 2018 state that “it is an offence for a person knowingly or recklessly to re-identify information that is de-identified personal data without the consent of the controller responsible for de-identifying the personal data”. The view of the law in relation to re-identification is covered in sections 171 and 172 of the Data Protection Act 2018 [94].

Arvind Narayanan and Vitaly Shmatikov [107] in their paper presented a new class of statistical re-identification attacks against high-dimensional microdata such as personal preferences, recommendations, and transaction records. A more predominant scenario of re-identification attacks involves those against microdata such as zip codes, age, gender and other individual-related variables. The paper analysed the application of a re-identification methodology to the Netflix Prize dataset, a dataset containing 500,000 subscribers. The article demonstrated that an attacker who is aware of a little bit of information about a subscriber in the dataset can identify the subscriber’s record from the dataset with limited difficulty. Therefore, using a third-party information source (in this case, reviewer information from the Internet Movie Database) as a means of acquiring background knowledge, the researchers were able to identify the Netflix records of users and able to discover these individuals’ political orientation and more potentially sensitive information. The authors concluded by reflecting on their re-identification methodology and how data holders are incapable of foreseeing possible computations that may be performed on the released datasets. The paper considered the possibility that re-identification of the Netflix Prize dataset would be harder (not impossible) if the dataset was released without the movie names. The authors find this countermeasure an interesting one to investigate. The countermeasure bears correlation to Samarati and Sweeney’s postulation in [130], expressing that anonymisation will work better if the most important attributes about individuals in an anonymised dataset do not belong to any quasi-identifier.

Addressing the issue of re-identification from a criminal and legal perspective, Salvador Ochoa et al [109] published a paper titled “Re-identification of Individuals in Chicago’s Homicide Database: A Technical and Legal Study”, exploring re-identification using crime datasets. A publicly released anonymised dataset containing information about the victims of homicide in Chicago over a thirty years span was analysed, this dataset was then cross-referenced against the Social Security Death Index database. 35% of the victims were associated with their details from the shared homicide dataset. The study illustrated the re-identification method and results and also included a legal review of the United States regulation pertaining to data re-identification.

The paper concludes by recommending that sensitive databases should be removed from online locations and that national re-identification regulations should be established [109].

[115] surveys approaches that have been proposed by publications for a criminal or penal sanction towards wrongful re-identification of anonymised datasets. The paper noted the increase in the call for criminal penalties for re-identification offenders, especially in the context of biomedical data; to allow the health sector to benefit from big data analysis. This call to criminalise re-identification hoped to convincingly address privacy concerns that may hinder medical innovations enabled by analysis and linkage of databases. However, the authors in [115] criticised the criminalisation advocates because their perspective focused on speculative opportunities rather than on important and practical advances that could be attained through big data analysis. The paper presented a review of various data re-identification criminalisation approaches that have been put forth. It questioned the need for specific rules prohibiting re-identification attacks, stating that explicit prohibition ideas are typically proposed without reference to data protection principles that already exist. This paper concluded by highlighting a suggestion that the path to reliable and lasting data protection enforcement is by empowering data protection authorities, in terms of their capacity, resources and jurisdiction to impose actions against unsanctioned re-identification. The authors proposed an abandonment of “complaint-based” enforcement for approaches that implements proactive investigation and monitoring. The work in this research aligns with and contributes to achieving proactive enforcement against re-identification attacks, as suggested by the authors of [115]. The paper also urged law and policymakers to provide appropriate rectification for when wrongful re-identification occurs by discouraging the behaviour and compensating the victims of re-identification attacks.

Esma Aïmeur, et al, in a 2012 paper [3] approached re-identification from the perspective of an attacker trying to reconstruct social media user profiles, from the information these users might have disseminated on the internet. The research was conducted to highlight the threat of profile reconstruction to individuals using social network platforms. The paper used Twitter as a case study, stating the high possibility of reconstructing Twitter user profiles strictly from information available publicly, even though Twitter is supposed to retain less personal information than platforms like Facebook. A new two-phase system, based on a method of data re-identification was proposed. The first phase involved searching for the subject’s (from a pool of 250 targeted Twitter user profiles, serving as the first dataset) information sprinkled over different websites online, including blogs and other social media networks, this is used to build a second dataset. The next phase involved creating a linkage between both datasets with the aim to construct a digital identity for the subject. According to the paper, the research was able to identify 41.6% of the owners of Twitter user profiles from the sample dataset [3]. The paper highlighted that individuals have a digital identity that is inadequately protected which leads to easy re-identification of such individuals. In conclusion, they draw a correlation between men and women with respect to how concerned they are with protecting their privacy. The work in this paper identified one in two men and one in three women, however, the authors are aware that their dataset included more male population and the work intends to expand its study to include more females. The paper noted a development it characterised as scary, around employees requesting Facebook login details (passwords included) from prospective employees. As a possible countermeasure, the authors advised and suggested having two Facebook accounts; one to keep impersonal and give employers and the other to use for real social connections. However, the authors also recognised and acknowledged that this is not foolproof.

In a paper published by Lukasz Kniola in 2016, [88], data re-identification is approached from a risk assessment perspective. The assessment was presented using a “quantitative approach”.

Using the context of any particular data release, the paper employed “metrics and methodology” to quantify the risks of re-identification and talked further about how to go about managing these risks; by figuring out the right level of anonymisation (de-identification) to be applied on the released dataset. Assumptions based on an estimation of the risk level that the dataset is exposed to after sharing are used by the researchers to determine and select tolerable risk level thresholds and ways to achieve this verification were also discussed in the paper. Two data release contexts were discussed, one of which includes a dataset released under a strict contract with the data sponsor stipulating and enforcing requirements and restrictions that must be adhered to by the data user. The other context mentioned in the paper is public release, where the dataset is released to the public domain, with no control and accountability on how it ends up being used. In the latter context, the paper proposes that conservative assumptions should be made about the risk level that the shared dataset is exposed to, as potential attackers can gain access to the redacted details of the dataset.

Furthermore, the probability of different re-identification attempts was identified in the form of categories, which are: deliberate attempt, inadvertent (spontaneous) attempt and data breach. The paper concluded that anonymisation is a probabilistic measure and it is never capable of eliminating the risks of re-identification to zero while retaining some level of data usability. Emphasis was made on how calculating the risks of re-identification of anonymised detail in a dataset using a quantitative approach is the most comprehensive way to perform due diligence, figuring out how subjective the database is to a re-identification attack (contextual risk assessments) in the process of anonymisation [88].

Approaching from a perspective that deviates from technical computer science or statistical literature, Mark Elliot et al [49] present their book to cater to the interest of organisations or government bodies that may have datasets to de-identify (anonymise). The book presents an Anonymisation Decision Framework (ADF), that can be employed (possibly with some slight alterations) to different dataset scenarios. The book presents the framework to be universally applicable, regardless of the data privacy regulatory jurisdiction the user is complying with. A term “*data situation*”, which describes the idea of treating the dataset and its environment as a total system was coined, and with this, the authors of the book view anonymisation as a heavily context-dependent process that needs to consider the data situation before concluding if and what kind of anonymisation is required. This means that the assessment and management of data re-identification risk should consider all the nuances of ADF, including the dataset, external data source, legitimate data usage and potential misuse, legal, and ethical responsibilities.

Similarly to [110], [49] also emphasises the misconception among data holders that ‘anonymised’ means no risk of sensitive data disclosure, it also debunks the “*release-and-forget*” mentality. The principles of the ADF involve two courses of action, which are categorised into technical and contextual. The technical allows quantification and management of re—identification risks, while the contextual element enables data holders to address the factors that affect re-identification.

The Information and Privacy Commissioner of Ontario [72] published a paper on the guidelines for the de-identification of structured data, comparable to [49], in the sense that it presents a framework approach to data anonymisation. The paper approached generating anonymisation guidelines from a risk-based methodology, involving the estimation of an acceptable threshold for re-identification risk for a given data release. The paper’s stipulation that an acceptable risk level has to be calculated for “a given data release” is in agreement with [49]’s “dataset and its environment” ideology, however, the calculation of re-identification risk presented by [72] considers different factors. These factors include whether an attacker knows if a target is in the

dataset, a situation where the attacker can tell the target is in the dataset is called “prosecutor risk”, and the attacker not being able to conclude that the target is present in the data is known as “journalist risk”. The paper presents a framework that will function based on a systematic analysis of the level and the kind of re-identification risk involved in the data release.

Yves-Alexandre de Montjoye et al, [33] explored re-identification using financial data, by analysing the re-identifiability of credit card metadata. The journal article emphasises how meta-data (data describing other data, serving as an informative label) that are generated by different daily technologies (location applications, entertainment and social media platforms) may reveal sensitive personal details about an individual. Financial data like digital payment details include rich metadata about the individual’s behaviour. To present an assessment of a possibility of re-identification from financial data, the researchers for this paper gathered dataset of three months of credit card information for 1.1 million users. The names, account numbers and other direct identifiers were removed from the dataset and every transaction was time-stamped. Using what the paper referred to as “*unicity*” (the risk of re-identification through having knowledge of outside information about an individual), the risk of re-identification was measurable. With the knowledge of when a user uses their credit card, the researchers were able to figure out a recognisable pattern. The paper concluded that, having knowledge of four random spatio-temporal data: data about the space (location) and time about an individual is enough to re-identify 90% of individuals from a poorly anonymised dataset. Similar to [3], the paper also draws a comparison in the re-identifiability of men and women, stating that women are more re-identifiable than men with credit card metadata. The paper suggested that the results from their work emphasized the need for both technical and policy-based countermeasures against re-identification.

Yves-Alexandre de Montjoye, et al, in another publication [32], “Unique in the Crowd: The privacy bounds of human mobility” further explore the relationship between re-identification and spatio-temporal data and found that human mobility traces are very unique. The paper collected fifteen months of human mobility data for one and a half million individuals. For the subjects in the dataset that has their location specified hourly and with spatial resolution equal to that given by the carrier’s antennae, four spatio-temporal points are enough to uniquely identify 95% of the individuals. The researcher in this paper then coarsens the data spatially and temporally (added noise to the location and time elements of the data) to find a formula for the uniqueness of the individuals’ mobility tracking, given their spatial resolution and other available information from external sources. The work shows that even coarse datasets provide poor anonymity and further represents the importance of frameworks, systems and institutions dedicated to the data privacy protection of individuals.

Work	Perspective	Proposed Solution
Samarati and Sweeney (1998) [130]	Anonymisation	k-anonymity
Sweeney (2002) [149]	Anonymisation	k-anonymity
Ohm (2010) [110]	Anonymisation and data release	Data privacy law (e.g. re-identification ban)
A. Narayanan and V. Shmatikov (2008) [107]	Data release and re-identification	Better anonymisation
Ochoa et al. (2002) [109]	Re-identification and law	Better anonymisation and national regulations
Phillips et al. (2017) [115]	Re-identification and law	Proactive investigation and monitoring
Aïmeur et al. (2012) [3]	Re-identification and social media	Multiple social media accounts
Kniola (2016) [88]	Data re-identification, data release and risk assessment	Contextual risk assessment during anonymisation
The Information and Privacy Commissioner of Ontario (2016) [72]	Anonymisation, data release, re-identification and risk assessment	Contextual risk assessment during data release
Elliot et al. (2016) [49]	Anonymisation, organisations and government bodies	Anonymisation Decision Framework and contextual risk assessment
de Montjoye et al. (2013) [32]	Re-identification and spatio-temporal data	Data privacy frameworks, systems and institutions
de Montjoye et al. (2015) [33]	Re-identification and financial data	Advanced privacy-conscientious technologies and quantitative assessment of re-identification likelihood
This research	Re-identification and database query data	Proactive detection of re-identification with a prototype state of the art data mining model

Table 2.1: A Comparison Review of Work in Anonymisation and Re-identification

2.2 Data Anonymisation

The data anonymisation procedure involves de-identifying details of a dataset such that the personal and sensitive information is not directly interpretable, anonymisation process replaces the actual data value with a different value while preserving the data format and type [120]. Data anonymisation measures alter and convert the data into a form that obscures the direct meaning of the data records, such that individual-specific information is not readable by users after the data is published.

To allow data to be published and shared for research and other data processing reasons, anonymisation measures are employed to make the dataset to be shared privacy-safe. Data anonymisation uses methods including generalisation, perturbation, pseudonymisation and suppression to present an individual’s data records as indistinguishable from other records in the dataset [1]. It involves the de-identification of datasets such that the data produced should not allow individual records to be correlated back to an entity, as the newly generated de-identified data does not include quasi-identifiers or translation variables to allow an attacker to make such correlation [108]. Anonymisation aims to prevent malicious users from inferring private or sensitive information from the dataset, however, the data should still be useful for processing by

honest data analysts. However, query patterns employed by attackers on anonymised datasets pose a credible threat to data privacy aimed to be achieved by data anonymisation. This research is focused on anonymised datasets and query patterns that attackers use to interact with the data systems holding anonymised data.

2.3 Overview of Data Anonymisation Measures

In an attempt to use data anonymisation to eliminate subject-specific information from publicly shared datasets, different measures have been developed and adopted to achieve this. Datasets generally have common attributes, based on the type of details contained in them. Different types of attributes in data collected about individuals are;

- **Direct Identifiers:** These are data attributes that are explicitly associated with the data subject (the individual that the data is about). E.g. names and NI numbers.
- **Quasi Identifiers:** These are attributes about the data subject, which can be combined with other information to form details that can be used to uniquely identify a data subject from an anonymised dataset. For example; age, postcode, gender etc. Quasi-identifiers are also referred to as *key attributes*.
- **Sensitive Identifiers:** These are data attributes that the data subjects are possibly unwilling to publicly share or let get associated with them. Examples include medical data and financial data. These are also known as confidential attributes.

Anonymisation measures includes k -anonymity, l -diversity, t -closeness and p -sensitivity, perturbation, pseudonymisation and differential privacy. This section presents the different techniques of anonymisation and explores data privacy from the perspective of each anonymisation technique, analysing these techniques in terms of their features and limitations. The different anonymisation measures operate with data stored in relational databases, presented in tables with rows (records) and columns (attributes) [22]. Anonymisation removes all direct identifiers from the database table. However, with the table clear of direct identifiers, other attributes still remain. These remaining attributes can be combined with other publicly available attributes to specifically identify an individual.

2.3.1 k -Anonymity

k -anonymisation therefore aims to alter a database, such that no user can infer a highly probable association between records in the database table and the corresponding subjects [22]. The k -anonymity measure requires that, for every record in the database, there must be at least $(k-1)$ indistinguishable other records in that database. Some attributes of records in the dataset are *generalised* or *suppressed* such that each record is indistinguishable from at least $(k-1)$ other records. To satisfy k -anonymity, a combination of the generalised or suppressed attributes in the dataset will not generate matches fewer than k individuals. The k -anonymisation criterion works with respect to the quasi-identifiers in the dataset, such that every combination of quasi-identifier values can be indistinctly matched with at least k record entries [112, 25]. A group of indistinguishable records as a result of k -anonymisation is referred to as *equivalence class*. An example of a 3 -anonymous table is represented in Table 2.3.

Post Code	Gender	Age	Diagnosis
LE1 9BH	Male	28	HIV
LE1 6TB	Male	30	Obesity
LE1 9BC	Female	26	Flu
LE1 6TZ	Female	27	HIV
LE1 9BQ	Female	27	HIV
LE1 6TW	Male	34	Flu

Table 2.2: Sample database I

Post Code	Gender	Age	Diagnosis
LE1 9*	Person	25-30	HIV
LE1 6*	Person	30-35	Obesity
LE1 9*	Person	25-30	Flu
LE1 6*	Person	30-35	HIV
LE1 9*	Person	25-30	HIV
LE1 6*	Person	30-35	Flu

Table 2.3: β -anonymous table for sample database I

The example in the tables above represents a k -anonymity implementation, where $k=3$ and Quasi Identifier (QI) = Post Code, Gender, Age. The anonymised version of the table of the dataset represented in Table 2.3 shows a β -anonymous dataset table, implementing the k -anonymity criteria to perform anonymisation on the dataset. Quasi-identifier attributes are used to de-identify the dataset such that at least β records are not distinguishable in relation to the quasi-identifiers. The table is β -anonymous because there are 3 records in each equivalence class. The attributes Post Code, Gender, and Age are quasi-identifiers, with which k -anonymisation is achieved.

Methods for Achieving k -Anonymity

The k -anonymity process adopts two methods to achieve data anonymisation, these methods are generalisation and suppression.

- **Generalisation:** This method involves the substitution of attribute values in the dataset with more generic values. The replacement attribute value is a less specific or defining one, but it is semantically consistent. An example of generalisation is shown in the value of the “Gender” and “Age” attributes in table 2.3.
- **Suppression:** In suppression, the values of records are simply not released. Suppression involves fully or partly redacting value of records. An example of suppression method in k -anonymity is shown in the “Post Code” attribute in Table 2.3.

Attacks Against k -Anonymity

As an anonymisation technique, the sole objective for k -anonymisation is to protect privacy in published data. However, the technique is vulnerable to attacks that may consequently lead to

data privacy breach. As described in [98], k -anonymity is subject to two possible attacks, namely homogeneity attack and background knowledge attack.

- **Homogeneity Attack:** This attack on k -anonymity takes advantage of any similarity in the sensitive identifiers contained in the dataset. It has been established that k -anonymity works in relation with quasi-identifiers, therefore, similarity is bound to occur in the sensitive identifier for the equivalent classes as the size of the dataset grows larger. A successful attack is possible, if an attacker knows the quasi-identifier value of an individual and is also aware that this individual is represented in the dataset, an attacker can therefore infer sensitive information [25]. As a result of homogeneity attack, the sensitive attribute of a subject in a k -anonymised dataset can be revealed as the dataset grows in size and sensitive attributes get repeated within an equivalence class.

The larger the size of the anonymised dataset, the lower the diversity of the sensitive data. This suggests that, in addition to k -anonymisation, the dataset should also ensure diversity, such that every equivalence class that shares the same quasi-identifier values should have diverse values for its sensitive data attributes.

- **Background Knowledge Attack:** The background knowledge attack in k -anonymity is predicated on the attacker having prior information from external sources about a specific individual represented in the dataset. In a scenario where the quasi-identifier values are the same and there is minimal diversity in the values of the sensitive data attribute, an attacker trying to infer private information will be able to make very close predictions, if armed with background information about the data subject. To combat these weaknesses in k -anonymisation, a new anonymisation criterion, called l -diversity was introduced.

2.3.2 l -Diversity

The flaws in k -anonymisation imply that an attacker can discover the values of sensitive attributes, when there is little diversity in the values of these sensitive attributes [16]. Also, the k -anonymisation technique is a relatively a weak privacy guarantor technique when the attacker has background knowledge about a specific scenario. A stronger anonymisation technique that takes diversity into account is required and this is where l -diversity comes into the fold.

The l -diversity principle states that an equivalence class is l -diverse, only if it contains at least l well-represented values for the sensitive attributes, and that a dataset table is considered l -diverse if every equivalence class is l -diverse [98]. l -diversity provides privacy regardless of the background information in the possession of the attacker. The core concept of l -diversity requires the values of the sensitive attributes to be well represented in each equivalence class [98].

	Non-sensitive		Sensitive	
	Post Code	Age	Nationality	Diagnosis
1	LE1 6TB	30	Sudanese	HIV
2	LE2 7GP	30	Sudanese	HIV
3	LE2 7GN	26	Armenian	Cancer
4	LE1 6AY	36	British	Cancer
5	LE4 9VN	41	British	Asthma
6	LE4 9HI	48	British	HIV
7	LE4 9VD	45	French	Cancer
8	LE4 9EN	50	Nigerian	Cancer
9	LE1 6GZ	31	American	Asthma
10	LE1 6FT	31	Italian	Asthma
11	LE2 7EU	25	Indian	Asthma
12	LE2 7AP	28	Pakistani	Asthma

Table 2.4: Sample database II

	Non-sensitive		Sensitive	
	Post Code	Age	Nationality	Diagnosis
1	LE1 6*	≥ 30	*	HIV
2	LE1 6*	≥ 30	*	Cancer
3	LE1 6*	≥ 30	*	Asthma
4	LE1 6*	≥ 30	*	Asthma
5	LE4 9*	> 40	*	Asthma
6	LE4 9*	> 40	*	HIV
7	LE4 9*	> 40	*	Cancer
8	LE4 9*	> 40	*	Cancer
9	LE2 7*	≤ 30	*	HIV
10	LE2 7*	≤ 30	*	Cancer
11	LE2 7*	≤ 30	*	Asthma
12	LE2 7*	≤ 30	*	Asthma

Table 2.5: \mathcal{I} -diverse table for sample database II

The concept of l -diversity requires that every equivalence class has at least l uniquely represented values for each of the sensitive attributes [95]. The l -diversity technique provides a significant improvement when compared to k -anonymity, particularly in the level of anonymity l -diversity offers. However, l -diversity also possesses limitations, including the fact that it may be difficult to achieve an l -diverse dataset table. For instance, an original dataset with only two values for the sensitive attributes; suppose the results of a medical test can either be positive or negative. If the instance is further assumed to contain 1000 records, with 99% of results being negative and 1% positive. The two values of sensitive attributes in this scenario have a very different level of sensitivity [95]. In this case, there will be equivalent classes that contain records with only negative values, and a 2 -diversity will be ineffective to keep the sensitive attributes diverse. To achieve a 2 -diverse table that has distinct values when large numbers of records are involved will eventually cause significant information loss. Similarly to k -anonymity, the anonymisation measures provided by l -diversity have weaknesses and is susceptible to attacks. The attacks used by malicious individuals to exploit l -diversity are *skewness* and *similarity* attacks, as described

below.

- **Skewness Attack:** In a health dataset where there are an equal number of patients with and without the same sensitive attribute value (e.g. a diagnosis), thereby creating a 2-a diverse group of sensitive attributes [40]. If an attacker can infer that a particular individual's record is in this dataset, that individual can be considered to have a 50% probability of having the sensitive attribute linked to them.
- **Similarity Attack:** Sensitive attributes in datasets can also get compromised in l -diversity by similarity attack. This involves an instance of sensitive attributes in an equivalence class being semantically similar, even though the equivalence class is l -diverse [125]. For instance, if a 3-diverse dataset has sensitive values that are close in meaning in the equivalent class, such as diagnosis entries of kidney cancer, lymphoma and melanoma; an attacker that can link an individual to that equivalence class can infer that the individual has cancer.

2.3.3 t -Closeness

The shortcomings of l -diversity brought about the need for a new privacy measure. t -closeness was introduced as a method to keep privacy when l -diversity fails. The principle of t -closeness states that an equivalence class is said to have t -closeness if the distance between the distribution of a sensitive attribute in the class, and the distribution of the attribute in the whole table is no more than a threshold t . A dataset table is said to have t -closeness if all the equivalence classes have t -closeness [95].

t -closeness attempts to overcome the privacy exploitation that l -diversity is susceptible to by having it such that, the distribution of sensitive attribute in an equivalence class is a mirror of the sensitive attribute distribution of the whole dataset, leaving no occurrence of semantic similarity in an equivalence class that does not occur in the entire dataset (although, similarity in an equivalence group will be unavoidable if all attributes are of similar nature, e.g. patients having similar diseases) [40].

An acknowledged shortcoming in the t -closeness anonymisation measure is its limited amount of useful information that can be produced, as applying t -closeness hinders any correlations between quasi-identifiers and sensitive attributes. The only way around this, is to increase the threshold t , therefore, taming t -closeness [40].

2.3.4 p -Sensitive

This anonymisation criterion is also referred to as p -sensitive k -anonymity, and a dataset can only satisfy p -sensitivity requirements if that dataset satisfies k -anonymity, and for every equivalence class with identical distribution of quasi-identifiers in the dataset, the number of distinct values for every sensitive attribute occurs at least p times within the same equivalence class [152]. To put it in context, Table 2.6 below illustrates a 4-anonymised dataset with respect to the quasi-identifiers: age, post code and gender. To determine the value of p , every equivalence class is to be analysed. The first equivalence class contains two different diagnoses and just one income, therefore the p value is 1 because that is the least times a sensitive attribute occurred in the class. This can be said to have satisfied 1-sensitive 4-anonymity property. The second equivalence class

	Age	Post Code	Gender	Diagnosis	Income (£)
1	25	LE1 6TB	M	Cancer	7,000
2	25	LE1 6TB	M	Cancer	7,000
3	25	LE1 6TB	M	AIDS	7,000
4	25	LE1 6TB	M	Cancer	7,000
5	40	LE1 9BH	F	AIDS	5,000
6	40	LE1 9BH	F	AIDS	10,000
7	40	LE1 9BH	F	Asthma	10,000
8	40	LE1 9BH	F	Asthma	5,000

Table 2.6: p -Sensitivity database table

having at least two distinct values for both the “Diagnosis” and “Income” attributes, would have a p value of 2.

In p -sensitive k -anonymity, the value of p is always less than or equal to that of k [152]. Therefore, to implement the p -sensitivity measure, a dataset must satisfy k -anonymity with the value of k greater than or equal to 2. Also, preventing attribute disclosure in the dataset will require a value of p greater than or equal to 2. In theory, a dataset satisfying these two stated properties (2 -sensitive 2 -anonymity) should be privacy-safe. However, an attacker has a 50% chance of guessing the identity or attribute of the individuals. This percentage chance gives an attacker an unacceptable advantage, therefore, the values of k and/or p must be higher [152].

2.3.5 Pseudonymisation

The process of pseudonymisation substitutes or eliminates direct identifiers in the dataset that directly relates to a natural person, either partially or completely, such that data subjects are unidentifiable [141]. In pseudonymisation, the data holder handles the dataset such that sensitive or personal data is not relatable to any specific data subject unless a secondary source is employed. The UK GDPR (General Data Protection Regulation) considers a dataset to be pseudonymised if any secondary data source that can be used to make data subjects personally identifiable is stored separately from the pseudonymised dataset [127, 18]. Pseudonymisation replaces direct identifiers in the dataset with *pseudonyms* that are randomly produced or worked out by an algorithm. This anonymisation measure replaces the PII in the dataset while preserving the utility of the dataset.

Pseudonymisation can be achieved with a number of techniques including *secret key encryption*: using a single key to encode and decode the information, *hashing*: converting and representing a piece of information of arbitrary length into a fixed length value, and *tokenization*: replacing sensitive records in the dataset with non-sensitive substitute values referred to as “tokens” [89].

The shortcomings of the pseudonymisation measure include the difficulty in accurately deciding the attributes to be considered direct identifiers and to be substituted in the pseudonymised dataset. Another weakness in pseudonymisation relates to when a predetermined algorithm is used to assign pseudonyms in the dataset. An attacker who is able to discover the algorithm used in altering the direct identifiers can reverse the pseudonymisation and re-identify the dataset. Also, a pseudonymisation scenario where the same distinctive pseudonym is repeatedly used in a dataset, in multiple data or over an extended period of time can be ineffective. The likelihood of re-identifying a subject by quasi-identifiers associated with the distinctive pseudonym is increased in this scenario [96].

2.3.6 Perturbation

The technique of data perturbation involves changing the values of the records in a dataset without altering the fundamental meaning of the data. “Noise” is added to the dataset for the purpose of keeping the details of data subjects anonymous. When perturbation techniques are implemented on a dataset, the aim is to simultaneously protect the dataset from a privacy breach and preserve the underlying properties that gives meaning to the data, thereby making it usable in the context of data releases [163, 100]. An instance of noise addition for data perturbation involves adding or multiplying a randomised number to personal or sensitive quantitative attributes in the dataset, to conceal the distinguishing values [105].

Data perturbation methods can be of two fundamental classes, which are *input perturbation* and *output perturbation*. The input perturbation technique involves random modification of the underlying dataset and outputs to queries or analysis on the dataset is generated using the modified dataset. The output perturbation method on the other hand, has the computation of an analysis or query executed against the original dataset (unmodified), however, a noisy version of the output is presented to the user of the dataset [42].

2.3.7 Differential Privacy

In response to the shortcomings that exist in different data anonymisation measures, a new privacy model was introduced known as differential privacy. Differential privacy is a formal mathematical framework for ensuring privacy protection when statistical data are to be analysed or released [165]. In differential privacy, an adversary with access to the output of analysis will learn roughly the same information whether or not a single individual’s data was included. Differential privacy makes almost no assumptions about the attacker’s background knowledge [30]. Differential privacy is neither a single tool nor a criteria, but a standard for quantifying and managing privacy risks for which several technological tools and techniques have been developed. These differential privacy tools are designed to find implementation in various sectors of academic, industrial, and government settings.

Differential privacy is a definition, as opposed to an algorithm. It is a definition geared towards the problem of privacy-preserving data analysis. To achieve differential privacy, data perturbation is utilised. This involves adding noise to the dataset to enforce confidentiality such that query results on the database return perturbed aggregated values (a differentially private output). The output from differential privacy is presented such that analysis on the database is unable to tell if a specific record has been altered or not, suggesting that an attacker cannot deduce any information about any record in the database from the perturbed output of a computation [105, 30].

A data analysis process is considered to have protected the privacy of the individuals in the dataset if the output of the analysis does not reveal any information about any particular individual. Differential privacy standardises this process as a mathematical definition, which can subsequently be used in designing a privacy-preserving analysis process that provides a mathematical guarantee of privacy protection. In differential privacy, privacy is not just a property of the output, but a property of the analysis and computation that generated the output [165]. Take into context a scenario whereby;

“a participant (Alice) who is aware that individuals have been re-identified in the past after released of de-identified data from a previous study, will be concerned that sensitive information

about her could eventually be revealed in the de-identified data release from the new study she is participating in” [165].

Differential privacy addresses Alice’s concerns in the scenario above. This is because if the analysis of the data from the study is designed to follow the differential privacy framework, then Alice is guaranteed that the output of the analysis will not disclose any details specific to her, even though her information was used in the analysis. Furthermore, considering a scenario whereby Alice decided to not participate in the study, and a differentially private analysis was performed on the study data with only the exemption of Alice’s data. Alice’s data being exempt from the analysis protects her privacy because the output of the analysis does not depend on her specific information. Also, the output of the analysis will not be affected if Alice’s data were completely different [165]. Differential privacy assures a user who is conflicted about participating in a statistical database study, that their participation will not significantly affect the output of the analysis performed on the database [153].

The Promise of Differential Privacy

Differential privacy promises to protect individuals from additional privacy vulnerabilities that might occur as a result of their data being in a statistical database: a database system that allows users to fetch aggregate values for a subset of the subjects in the database, aggregate values including mean average, count, and sum values [155]. It offers privacy protection against vulnerabilities that would not have applied to these individuals if they had not participated in the statistical database. Differential privacy promises the probability of being vulnerable is not significantly heightened by an individual’s choice to participate. With respect to possible future harm, differential privacy sees to it that an individual can be indifferent in the decision on whether or not to participate in a database survey [43].

Limitations of Differential Privacy

Differential privacy does not assure unconditional freedom from privacy leaks, as there is no guarantee that an individual’s secret will remain secret. Differential privacy simply guarantees that neither an individual’s participation in a statistical database will be divulged, nor will there be any disclosure of any personally specific information contributed, as a result of the individual’s participation in the statistical database.

Features of Differential Privacy

With the definition and mechanisms of differential privacy described, there are properties of the privacy model that are to be understood. The key features of differential privacy include:

- Differential privacy does not only protect against re-identification but also protects against arbitrary privacy risks.
- It performs automatic neutralisation of linkage attacks (matching anonymised entries with non-anonymised entries from a different dataset). Satisfying differential privacy is a feature of the data access mechanism, and is unrelated to the presence or absence of a secondary source of information available to an attacker.

- Quantification of privacy loss. Differential privacy has a measure of privacy loss, permitting comparison among different techniques with respect to the accuracy and privacy provision of these techniques.
- Closure under Post-Processing. Differential privacy is resistant to post-processing, making it impossible for a data analyst with no additional knowledge about the database to compute a function of the output generated by a differentially private algorithm, and make it less differentially private. This implies that, privacy loss cannot be increased through analysis of the data by the analyst thinking about the output of the algorithm; regardless of the additional information available [43].

The different data anonymisation measures as described in this section, all target Privacy Preserving Data Publishing (PPDP) [26]. These techniques become necessary when data holding bodies wish to release a set of data they are holding about various individuals to the public, opening the dataset to all sorts of analysis and queries. Alternatively, differential privacy leans more toward Privacy Preserving Data Mining (PPDM). In PPDM, the query to be answered must be known prior to the application of the privacy-preserving algorithm. This is not the case in PPDP [26].

2.4 Data Re-identification

Re-identification is the process whereby the data of the subjects in an anonymised dataset becomes distinguishable and can be matched with the identities of these subjects, this can occur as a result of insufficient or poor anonymisation, the attacker having prior information about the features of a data subject and an attacker inferring sensitive details about an individual from the properties of a released dataset. The goal of anonymisation is to prevent re-identification, therefore, an attempt to perform re-identification on anonymised datasets is referred to as a re-identification attack. There are different motivations for performing re-identification by individuals and organisations. These motivations include testing the quality of anonymisation in the dataset, gaining bragging rights or professional standing for performing the re-identification, causing harm and embarrassment to the organisation that anonymised the dataset, obtaining direct benefit from the re-identified dataset, and harming or humiliating the individual whose sensitive data can be learned as a result of the re-identification [55]. Over the years, there have been instances of successful re-identification attacks on anonymised databases, as reported in the academic literature [96, 107, 109, 143]. This has weakened the level of reliance that is placed on anonymisation techniques because in each case the databases were believed to be privacy-protected before they were shared publicly.

It is difficult to measure the re-identification risk of a dataset, considering that the chance of a successful re-identification depends on the quality of anonymisation of the original dataset, the technical know-how of the attacker, the resources an attacker has at their disposal, and the existence of additional data that can be linked with the subjects of the anonymised dataset. Generally, the risk of re-identification will be heightened as a result of improvement in attackers' skills and techniques, increasing computational power, availability of sophisticated tools and additional information becoming available about the individuals. Computing and reporting re-identification risk likelihood will typically involve a scenario that describes the rate of success and an assumption of the attacker's resources and skill level.

With the rapid advancement of modern information technology solutions, relational Database Management Systems (DBMS) used to store valuable and sensitive datasets, provides standard database security measures including encryption, access control, authentication [67]. However, in the context of data sharing, these security measures do not provide protection against data exploitation by malicious users aiming to re-identify anonymised database subjects by linking the anonymised dataset with other publicly available databases with high likelihood of having the same subjects in them.

Anonymised data are typically re-identified by combining two or more databases, in search of fragments of information that may reveal that the information from these databases is about the same individual. Another way re-identification of anonymised datasets can occur is by having closely related information about the same entity stored multiple times (data redundancy). When the process of re-identifying an individual from an anonymised dataset is performed successfully, it has dire data privacy protection repercussions, as it may violate the conditions under which the information was divulged by the data subject, collected, and shared by the data holder. The goal of re-identification is to link information hoped to be anonymous to the data subject. Even though PII is the usual target, “other information” (quasi-identifiers) about the data subjects which are presumed not to be personally identifiable can be used to re-identify subjects from anonymised datasets [76].

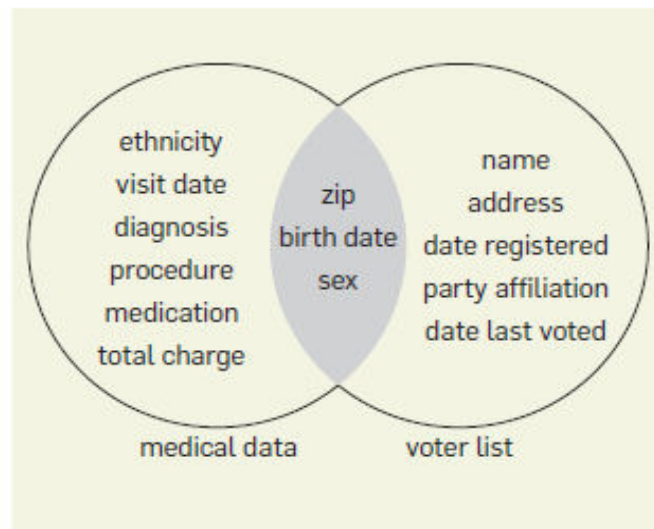


Figure 2.1: Joining two databases with redundant records for re-identification [7]

Legally, the European data protection law GDPR [70] has a strict definition of “anonymised”, requiring re-identification of the dataset said to be anonymised to be impossible. Pseudonymisation is seen as a security measure, and presumably by extension so are other weak forms of de-identification. The UK 2018 Data Protection Act [59] gives such security measures a special status by making re-identification illegal, with exemptions for research. According to [55], different re-identification scenarios include:

- Re-identification risk having to do with an individual in a dataset, who can be re-identified because the attacker is aware that such an individual is in the dataset. This is referred to as *prosecutor scenario*.

- Risk having to do with the existence of at least one individual in the dataset who can be re-identified. The attacker’s goal in this scenario is to prove that a subject in the dataset can be re-identified, and this is typically used to discredit the body behind the anonymisation. Referred to as *journalist scenario*.
- Risk having to do with the percentage of individuals in the dataset that can be accurately re-identified, this is known as the *marketer scenario*.
- Risk involving being able to differentiate between when an analysis is performed on a dataset with a re-identifiable subject and when the same analysis is performed on a dataset that does not include a re-identifiable subject. This is called *differential identifiability scenario*.

The aim of anonymisation is to allow a certain level of usage of dataset with sensitive records while simultaneously providing privacy protection by obscuring the identity of the data subjects. Interestingly, this aim presents a challenge that is difficult to address, as it involves a compromise between the level of anonymisation and usability of the resulting dataset. Regardless, anonymisation is necessary because it allows new uses for datasets that would have been forbidden under data privacy protection regulations. Therefore, it is the responsibility of data controllers, standards organisations, lawmakers and regulators to coin appropriate metrics to measure and determine an acceptable level of privacy security to create a balance between data anonymisation and data usability.

Public use of anonymised datasets can occasionally put the data subjects in a privacy-vulnerable situation. The risk of having this situation can be divided into three categories identified: *identity disclosure*, *attribute disclosure*, and *inferential disclosure*.

1. **Identity Disclosure:** This occurs when an attacker is able to correlate a specific data item to a specific individual. Identity disclosure can occur as a result of different scenarios, which include: insufficient anonymisation, re-identification by linking and pseudonym reversal.
2. **Attribute Disclosure:** This occurs when a piece of sensitive information can be attributed to a subject in the dataset. The occurrence of attribute disclosure does not depend on identity disclosure. For example, when the dataset shows that all individuals with common features have a particular attribute, and if the attacker is aware of an individual in the dataset with such features.
3. **Inferential Disclosure:** This involves a scenario when information can be deduced with high probability from the statistical properties of a released dataset. For instance, a dataset may present a correlation between income and the price of a car. Prices of cars are usually publicly available information, therefore, an outsider might be able to infer the income of a subject in the dataset.

2.4.1 Objectives of Re-Identification Attacks

Different de-identified databases reveal a variety of potentially personal information if re-identified. Depending on the motive of the attacker, the objectives of re-identification attacks may vary. A paper published as part of the work conducted in this research proposes the following characterisation re-identification attack objectives [76]:

1. **Universal Re-identification:** This is the re-identification objective when the attacker is aiming to re-identify everyone in the dataset. An attacker with this objective has no targets in mind, it is a penetration test on the anonymised dataset to determine the vulnerabilities in the anonymisation measures applied to the dataset and to explore how they can be exploited. The attack is looking to identify as many subjects as possible, therefore, its effectiveness is measured in success rate. The universal re-identification objective proposed in this research work has a correlation to the marketer re-identification scenario described in [55], as presented in section 2.4.
2. **Existential Re-identification:** This describes a re-identification attack scenario where the aim of the attacker is to re-identify at least one subject from the anonymised dataset, thereby proving that a re-identifiable subject (any subject) exists in the dataset. Re-identification is measured as either a success or a failure in this instance. This objective can be related to the journalist scenario of re-identification.
3. **Targeted Re-identification:** The re-identification attack in this case focuses on a particular subject in the dataset. In this scenario, the attacker already has an inclination that the targeted subject is in the dataset or may have one particular entry in the anonymised dataset that they want to re-identify. All re-identification strategy is directed towards identifying that specific individual. Targeted attacks are a special case of existential attacks, as the success rate of a targeted attack can be reported as an absolute yes/no or as a probability as in the Imperial College tool [128]. The targeted re-identification attack correlates to the prosecutor scenario.

More generally, other than a full targeted re-identification attack, another outcome can also be considered: where the identity of the individual in a database is not fully discovered, but other attributes are found out. For example, a particular medical diagnosis may reveal the gender or age of a patient. This is another characterisation of attribute disclosure and can be referred to as a *re-attribution attack*.

2.4.2 Overview of Data Re-identification Attempts

Depending on the level of privacy and security measures in the data release context and agreement or lack thereof, the motives, technical capability of the data recipient, and access to resources, the probability of a successful re-identification attempt varies. Performing an estimation of the probability of a re-identification attempt is most relevant when the dataset in question is released under a predetermined, contractual agreement, where the receiver of the dataset is recognised. In this scenario, actions can be put in place to assure that the dataset is used under the rules of the data sharing agreement. On the other hand, when data is released publicly, the probability of a re-identification attempt is presumed to be 1 (100%) [88]. Different literature [88, 72] have specified classifications of re-identification attempts, which are: *deliberate* (insider) attempt and *inadvertent* (accidental) attempt.

Deliberate Attempt

In this scenario, a potential attacker has a specific data subject(s) from the dataset in mind as a target (targeted/prosecutor attack). The likelihood of a data recipient of a non-public

dataset release attempting re-identification on a dataset is contingent on two factors [72], the first being based upon how extensive the predetermined controls and measures are (regarding the security and privacy of the dataset in the data release), and the second factor is the motives and capability of the data recipient (potential attacker) in regards to performing a re-identification attack [88, 72].

1. **Security and Privacy Measures:** The likelihood of a re-identification attack by the recipient of a non-public data release varies based on the level of security and privacy measures set in place in the data release agreement. The more advanced the level of the security measures are, the lower the probability of a re-identification attack being successful. Some employable security measures include:

- Data recipient must implement access control and least privilege policy, such that the dataset is only accessible by authorised users and for legitimate purposes.
- A non-disclosure contract must be drafted and signed by all the staff of the data recipient company involved in the data release. The contract serves as a pledge of confidentiality for all parties involved, including the staff, collaborators and subcontractors.
- There should be a maximum data retention period, after which the dataset should no longer be in the custody of the data recipient.
- Sharing data with parties not included in the release agreement should not be allowed without going through the procedures that are in compliance with the agreed security and privacy measures, such as getting the required authorisation.
- Transmission of the dataset through any digital media should be encrypted and documented with detailed chain of custody tracking.
- Anti-malware measures must be set in place for the protection of the dataset against malware infection.
- The hardware holding the dataset must be in a secured physical location, protected with sophisticated physical security measures [72].

2. **Motive and Capability:** The motivation and level of skill of the data recipient contributes to the probability of the recipient attempting a re-identification attack on the dataset. The more motivated and capable the recipient is, the more likely it is for an attack to be tried. Evaluating the motives and capability of a potential attacker (data recipient) will involve taking some factors into consideration, factors including:

- Whether or not the recipient individual or organisation has had a working relationship with the data releasing party in the past without any incident.
- Whether or not the recipient has authorised access to other private datasets that have been linked to re-identification.
- Whether or not the recipient has the technical know-how and other resources (e.g. financial) to attempt an attack.
- Whether or not there is a prevailing motivation (financial, bragging rights, grudge) for the recipient to attempt to re-identify one or more of the subjects in the released dataset.

A calculation founded upon the quality of the security and privacy measures employed in the data release contract, combined with the motive and capabilities of the data recipient could be generated to determine the likelihood of a deliberate re-identification attempt being perpetrated (usually from an insider attacker) [72].

Inadvertent Attempt

An attempt at re-identification is regarded to be inadvertent when the data recipient analysing the released data happens to recognise one or more individuals from the dataset. The dataset may contain a family member, friend or colleague that ended up being recognised by someone with access to the dataset (recipient). The probability of a re-identification attack occurring, in this case, depends on the data recipient having enough background information about a subject in the dataset to be able to identify them based on the available details in the dataset [88, 72].

2.4.3 Rationales for Data Re-identification

Different cases of successful re-identification attacks have proved how easy re-identification could be, under the right circumstances. However, like most other cyberattacks or data breaches, there is usually a motivating factor behind data re-identification attacks. Therefore, the question; “who would re-identify?” is always an angle to consider when addressing re-identification. There are quite a few individuals or bodies that would be inclined to perform a re-identification attack on anonymised databases. Different motivations for performing re-identification are discussed in this section.

Research

One of the major reasons data is collected and shared in the first place is for their use in scientific or statistical research. Although, the shared data may be anonymised, and the researchers are expected to work on this anonymised version, however, while trying to build and test theories, these researchers may want additional information about the data subjects and build a bigger profile in order to accomplish the aim of the research in question. An instance of this will be, a medical researcher studying anonymised health data, with generalised and suppressed properties of the subjects’ medical histories. Assuming the research finds unusual or intriguing links in the data about some of the subjects, then more information about these interesting subjects could be very beneficial to the success of the research. This could be a motivation for a researcher to want to perform re-identification of an anonymised dataset, even though, the data subjects may not want more information in the dataset to be discovered or worse - linked to them.

Reporting

Investigative reporters may analyse anonymised datasets with the goal of linking personal information that are embedded in them to an individual these information is about. In this case, the subjects of the information are usually public officials and celebrities. Investigative reporters may exploit (re-identify) a published de-identified dataset in order to report on the discovered information. Latanya Sweeney in [148] covered a scenario for this, with the anonymised health data shared by the Group Insurance Commission (GIC).

Marketing

In recent years, marketing has been an influencing factor in the rapid increase in data collection by various marketing institutions across the world. The quest for marketers is to generate holistic profiles about each of their customers, so as to be able to improve marketing strategy and implore a more direct and specific approach to their marketing. There is a lot to be gained by a marketing organisation looking to re-identify a customer database, because that would place such organisation in a position to exploit this information to improve their services and also sell the re-identified data to other service providers.

Insurance

Insurance companies have some obvious motivations for attempting to perform re-identification. Specifically health and life insurance companies, this may be a reason for the high level of attention being paid to the threat of re-identification in the medical field [109]. Further discovered (re-identified) information about an individuals' medical history, from an anonymised health dataset could be used by insurance companies to deny or increase the cost of insurance for certain individuals.

Blackmail

Celebrities and public officials make interesting and somewhat easy targets for re-identification attacks, because of their public profiles. In this instance, certain information is already known by the attacker because it can be publicly accessed. When re-identification of such individuals occurs, there is a possibility that the attacker may threaten to make the discovered information public, if some sort of demands are not met by the individual.

Politics

Political motivation and activism could also be another reason for attempting to perform re-identification attacks. For example, an anti-abortion group will be interested in re-identifying a dataset that could possibly reveal the identity of women who have had an abortion. This will be an instance of existential or targeted attack (journalist or prosecutor) as discussed in section 2.4.1. The political instances of re-identification could be a particularly sensitive one, because of its association with strong sentimental beliefs. These beliefs that may lead to the re-identified data subject being exposed to harassment or other kinds of harm.

2.5 Chapter Summary

In this chapter, different published literature with perspectives related to elements of this research is reviewed, looking closely into works in the data anonymisation and re-identification domain. This chapter helped the research reach the conclusion that, there is an overrated confidence placed in the ability of anonymisation measures to protect the privacy of subjects in shared anonymised databases, a realisation that was highly emphasised in [110, 149]. Also, literature [109, 3, 88, 32]

presenting the re-identification perspectives more specifically was explored to fully present the threat landscape that data re-identification is capable of posing on the safety of personal data.

A conceptual overview of data anonymisation was presented in this chapter, covering the various measures (including pseudonymisation, k -anonymity, l -diversity, t -closeness, p -sensitive, perturbation and differential privacy) that have been developed in the data privacy field and how they are implemented. The advantages and the drawbacks of anonymisation are also emphasised. Data re-identification was reviewed to continue presenting all the important concepts of this research. Different objectives and motivations of malicious users that attempt re-identification attacks were also explored. This research perceived the objectives of perpetrators of re-identification attacks to be: universal, existential and targeted.

Chapter 3

Data Mining and Database Query Pattern Recognition

The information age has spawned a world where massive amounts of data are collected on a daily basis. These data need to be analysed in order to discover any form of knowledge, by learning interesting patterns from the collected data in various applications. The Internet, other computer networks, and numerous other digital data storage platforms have data written into them on a substantial scale. These platforms are owned and maintained by organisations across various sectors including business, science and engineering, medicine, education, research, and information technology. The availability of data in abundance and the lack of potent data analysis techniques and tools creates a data-rich but information-poor environment. Therefore, automated and versatile tools and techniques are required to discover valuable information from the incredibly large amount of data, so these data can be transformed into organised knowledge. This requirement led to the conception of data mining and machine learning [62]. This chapter explores the concept of data mining and its implementation, setting the stage for selecting and applying different machine learning algorithms to the dataset in this research.

3.1 Data Mining (Machine Learning)

Data mining and machine learning are generally used to refer to the same concepts, as they both use the same models and techniques to achieve their purposes. However, a difference can be made in the context with which the two terminologies can be most appropriately used, this difference is based on the scope of the terms [91]. Data Mining is the process of extracting embedded, initially unknown and possibly useful patterns from a great deal of raw material (dataset) [62, 146]. It involves inferring knowledge from the analysis of a huge amount of dataset [62]. The goal here is to discover new information from a large dataset, information that is not obvious prior to the mining process. This assists data scientists in understanding complex connections between data. Data mining is focused on solving real human problems. For example, music streaming platforms will process the data about the music listening habits of their subscribers and use the knowledge discovered from the mining (processing) to predict and offer music options the platform now knows specific subscribers prefer. Data mining achieves this by studying, understanding and applying features of algorithms developed using machine learning.

Data mining is also referred to as Knowledge Discovery from Data (KDD), it is an interdisciplinary field that combines techniques from machine learning, pattern recognition, statistics, databases, data visualisation, and Artificial Neural Networks (ANN) [146]. Traditional techniques for data analysis allowed valuable data interpretations and important perceptions of the meaning of the data. These interpretations and perceptions are the knowledge that database builders and owners are ultimately after, however, in traditional data analysis, such knowledge has to be inferred by human data analysts and not automated tools. The amount of available data is too large for human analysts. Thus, in an attempt to overcome this limitation and satisfy the need for automated data analysis tools, experts in the field resorted to methods and concepts developed in machine learning.

Machine learning focuses on the development of algorithms and techniques that allow computers to be able to learn and recognise patterns automatically. There is no requirement for explicit programming for the functions that a machine learning system performs. The focal point of machine learning research is on discovering and demonstrating the mathematical properties of new algorithms [91].

The goal of data mining is to extract desired, meaningful and usable data from large amounts of databases and data mining has applications in various aspects including, customer relationship management and market analysis. During the cycle of data mining operation, from the gathering of data to knowledge discovery, sensitive details such as personal, financial and medical information are bound to be revealed as a result of the data mining process. The information and knowledge discovered by various data mining techniques may contain private information about individuals and businesses [110]. Therefore, preserving the privacy of the subjects whose data is being mined, has been one of the most prominent concerns in data mining [117], however, this is not the focus of this research. This work will be using data mining on database queries in order to protect data privacy.

3.2 Overview of Data Mining Techniques

The need for the analysis of large amounts of data with the aim to discover meaningful patterns and infer knowledge brought about the concept of data mining. To achieve data mining, several techniques have been developed for the discovery of knowledge from raw datasets. The different categorisation of data mining techniques is described in this section.

3.2.1 Predictive Data Mining

The predictive data mining task uses specific variables or values in the dataset to calculate unknown or future values of other variables of interest. It is a statistical technique to predict unknown outcomes. Different techniques employed in predictive mining include the following:

Classification

This is the most generally applied data mining technique and operates by employing a set of pre-classified examples to develop a framework that can classify the population of the whole dataset, these pre-classified examples are used by the classification algorithm to determine the parameters required for the prediction of a new data instance to a class. Classes or categories are

a collection of similar yet non-identical data samples grouped together based on their common features. Application of the classification machine learning technique is particularly suitable for data analysis tasks such as fraud detection, credit applicants' risk level, and spam email detection. The process of this technique involves learning: at which point the dataset gets analysed by the classification algorithm [15, 99]. Some algorithms used in classification include Decision Tree Induction, Naive Bayes Classification, Neural Networks, Random Forests, Logistic Regression, and K-Nearest Neighbors. [15, 52].

Regression

Regression is a supervised mining function for predicting a numerical target [52]. The regression model can be used to model the connection between one or more independent and dependent variables. Independent variables are those attributes that are already known and dependent or response variables are those that are to be predicted. However, many practical problems are not easily predictable. For example, stock prices, sales volume, course of resources consumption, and product failure rates are challenging to predict because they may depend on complex interactions of multiple variables [15, 52]. In these complicated cases, the implementation of more sophisticated techniques (such as logistic regression, decision tree, and neural network) may be required to predict future values. Since real-world practical prediction involves the combination of many complex attributes, different mining models have to be applied to implement prediction [99, 52]. Different regression methods include: Linear Regression, Multivariate Linear Regression, Nonlinear Regression, and Multivariate Nonlinear Regression [15, 52].

3.2.2 Descriptive Data Mining

Descriptive data mining techniques examine past events in the dataset to infer insights on how to approach forthcoming events. These techniques understand past performance through the mining of previous data to search for the reasons behind the success or failure of previous analysis [52].

Clustering

Clustering involves the identification of similar objects in the dataset and grouping these objects together into clusters. Unlike classification where pre-classified examples are used, the clustering technique adopts a different learning method where the classification of the example dataset is not pre-determined; however, some clustering models use both [52]. The implementation of clustering techniques can further identify both the dense and sparse areas in the object space and discover the complete distribution pattern and correlations in the data attributes. The clustering technique has found application in processes such as: grouping customers based on their purchasing patterns, categorising genes with similar functionality, and grouping related documents in emails [15, 99, 52]. Different types of clustering techniques include Partitioning methods, Hierarchical Agglomerative (divisive) methods, Density-based methods, Grid-based methods, and Model-based methods.

Association Rules Mining

The association rules mining technique is an approach for discovering the relationships between variables in large datasets. For example, in a set of transaction data, association rules mining

explores and discovers rules that forecast the presence of an item that depends on the existence of other items in the dataset. Practically, this mining technique may be used to direct the positioning of products inside a store to increase sales, can be used to examine logs on web servers to infer information about visitors on the company website, to study biological data with the aim of discovering new correlations among attributes in a dataset [52]. Different types of association rules mining techniques includes Frequent Patterns Growth and Apriori.

Anomaly Detection

This refers to the process of discovering patterns in a dataset that do not conform to the expected behaviour [23]. The anomaly detection process is suitable for detecting outliers or anomalies: which are samples of data that are significantly different from the rest of the dataset. The anomaly detection technique is applied in credit card fraud detection, telecommunication fraud detection, network intrusion detection, insurance, health care, and fault detection. The process of this technique builds a pattern summary as the basis of reference for “standard” behaviour to identify anomalies when there is a deviation from this standard pattern [52]. Anomaly detection is important in data mining because any anomaly noticed in the dataset translates to noteworthy, usually critical actionable information that is relevant to a variety of domains [23].

3.3 Overview of Data Mining Tools and Evaluation

Data mining and Knowledge Discovery in Databases (KDD) stays an evolving topic, and with commercial developers contributing to the rapid advancement of mining tools and technologies, the application of these technologies has been directed toward business and scientific problems. The increase in the demand for mining tools from different businesses and organisations brought about the development and availability of data mining technologies. This availability presented the task of deciding which data mining tool is the best for specific business and organisational needs and budgets. Choosing the wrong tool for a particular application could be costly financially, however, the cost in terms of finance is insignificant when compared to the cost in personnel resources, time consumption and the chance of making actionable decisions based on misleading data mining output [27].

There are bodies responsible for producing standardised frameworks for the development of data mining software, a well-known example is Two Crows Corporation [27]. However, this standardisation gets complicated because commercial data mining software vendors constantly update to newer versions of their software products, making the initially produced framework obsolete. Ultimately, the existence of a standardised framework has helped in deciding the important element to consider when evaluating data mining tools.

3.3.1 Data Mining Tool Evaluation Framework

Using commercial datasets from a variety of industries, [27] developed and proposed a methodology for selecting from the various available mining software tools. The methodology was predicated on first-hand experiences in data mining using real industrial datasets. The case study suggests four categories of criteria for evaluating data mining software tools. The categories are performance, functionality, usability, and support of ancillary activities.

1. **Performance:** This is the ability of a data mining tool to be able to handle a variety of data sources efficiently. The hardware configuration of the system running the tool has a major impact on the performance, from a computational perspective, even though different data mining algorithms are fundamentally more efficient than each other. The performance category focuses on the tool's ability to handle data under various situations as opposed to focusing on performance variables that are resulted from inherent algorithm features or hardware configurations.
2. **Functionality:**The functionality of the data mining tool has to do with the assessment of how well the tool adjusts to different data mining problem domains while considering whether or not the tool provides a sufficient variety of mining techniques and algorithms [27]. The tool should be adaptive and flexible, supporting customisation for both general and specific use. The functionality of a data mining software tool should also consider interoperability with other applications and the maximum number of concurrent users that can be connected and served by the system [16].
3. **Usability:** The data mining software tool should be capable of accommodating different levels and types of users without losing its functionality or effectiveness [27]. However, there is a shortcoming with easy-to-use data mining tools, because of the potential of it being misused, because it gives low-skilled, unethical and unprofessional users a chance to easily mine data. Unethical mining of datasets containing sensitive and personal records can potentially lead to breach of privacy against the subjects of the dataset. Nevertheless, a good mining tool should be easy to learn and have the capability to guide the user toward a proper data mining process.
4. **Ancillary Task Support:**The software should allow the user to perform data cleansing, manipulation, value substitution, filtering, randomisation, visualisation and other auxiliary tasks that support the process of data mining [27].

In the process of achieving the objectives of this research work, data mining is intended to be used in the analysis of the SQL query patterns, used to search and filter information on a database. Employing data mining techniques, queries with similar characteristics suspected to be attempting re-identification will be used to generate a baseline of what re-identification attack query patterns look like. With this, data mining techniques and appropriate machine learning algorithms can be implemented to train a system that can automatically recognise attempts of re-identification attacks.

3.3.2 Data Mining and Python Implementation

The use of the Python programming language and its extensive data science libraries for data mining and machine learning has become more prominent than the other available options in the field of data science [145]. Python libraries include built-in modules that allow access to the system's functionalities, such as file I/O (Input/Output) and modules that provide established solutions to a range of everyday programming problems, as well as more advanced problems [118]. These libraries include functionalities for data preparation, data visualisation, deep learning and machine learning. This research will explore Python and its libraries for the analysis of the research dataset (SQL queries) and the development of a system that can intelligently recognise re-identification attempt query patterns on anonymised databases.

Python frameworks used in this work to implement different machine learning algorithms are the Scikit-learn and Keras frameworks. *Scikit-learn* is an easy-to-use library that is capable of implementing various machine learning algorithms. *Keras* is a sophisticated deep learning API (Application Programming Interface) used for building and training artificial neural networks [56]. These frameworks include a vast amount of *modules* (Python files containing codes that can be imported for use in another Python program) and *functions* (Python codes in a module, that implement an algorithm). When implementing and training many machine learning algorithms, the configuration of hyperparameters is important to the performance of the resulting model [161]. The *parameters* of machine learning algorithms are derived during training, parameters inform the model about what to do with the dataset features. However, many machine learning algorithms have parameters that must be specified before they are trained. These types of parameters are referred to as *hyperparameters*, they provide the model with information about how to derive its parameters. During training, hyperparameters are tuned to achieve optimal performance from a machine learning model.

3.4 Review of work in Data Mining and Database Query Recognition

To the best of this research's knowledge, there is no published literature exploring the recognition of data re-identification attacks with respect to the patterns in database queries. This section presents work that has been published around database queries and the application of data mining techniques, drawing any parallel that exists between existing literature and this research work.

Kamra et al [83] in their work about detecting anomalous access patterns in relational databases, proposed a system geared towards intrusion detection functionality within a Database Management System (DBMS). The authors approached the development of the anomalous detection system by mining SQL queries stored in database audit log files. Profiles of what constitutes normal database access behaviour were modelled with data mining techniques. Access behaviour contradicting the normal profile is then identified as an intrusion. The work in this paper explored two scenarios. The first one is Role Based Access Control (RBAC), where normal profiles were modelled based on user permissions with respect to their roles. The intrusion detection system then determines role intruders, based on access behaviour expected from a user holding a particular role. The system is expected to recognise anomalous patterns in the activities of users that while holding a particular role, behave differently from their profiled behaviour. The work pointed out the benefit of this approach in detecting insider threats. The authors addressed the same problem without user role definitions in the second scenario presented in this work. According to the paper, this approach was considered because RBAC will be inefficient for systems with a large number of users. Clustering data mining techniques were explored for this scenario.

The work presented in [83] bears similarity with this research with its RBAC approach. This research is concerned with recognising re-identification patterns in database queries. Similar to this work, Ashish Kamra, et al, used query audit logs to build and maintain query profiles to represent consistent user access behaviour on the database. The work recognised their RBAC pattern recognition as a classification problem and they implemented a Naive Bayes classifier as a solution to this problem. Justification of the use of Naive Bayes points to its competitive performance and useful practical application in fields such as text classification and medical diagnosis. The result of the experiment in this paper reported a low false negative rate, showing

that the Naive Bayes classifier rarely confused an anomalous access pattern on the database as normal. However, the authors expressed concern over the high false positive rate of the model. They assumed it could be a result of the specific nature of the dataset.

Similar to [83], the work in [78] proposed a fully automated Intrusion Detection System (IDS) on a database, the detection implements RBAC to recognise anomalies in database queries. The database in this work stores executed SQL queries in a new data structure called Octraplet (an eight-array data structure). The new data structure proved advantageous for the work in this paper because of its efficiency, thereby improving performance regardless of the dynamic changes in the size and structure of the database. The authors emphasized that their system is flexible and can be adjusted to fit to function of databases. Naive Bayes classifier is implemented in the database Intrusion Detection System, to detect queries that are outside of the scope of what constitutes normal query behaviour on the database. Attacks such as privilege escalation and unauthorised privilege abuse executed against a database by both insider and outsider attackers are addressed by the system developed for the work in this paper.

The system proposed by Jayaprakash, et al, in [78] functions using four components, the database log files, a profile generator, a comparison mechanism and a response engine. The database log files are used to generate normal query profiles and the Naive Bayes classification algorithm is used to learn these profiles from logs. The profiles are stored for later comparison. When users issue queries on the database, an Octraplet is generated and a role profile gets created based on the Octraplet. The newly created query profile is then compared to the previously stored normal profiles. If the match is positive, suggesting that the issued query fits a normal profile, the query is executed on the database. Using a 1 to 10 scale, an alarm notification is sent to the administrator if the query has an anomaly rate above an 8.0 score. A query with a score between 6.0 and 8.0 is automatically blocked, queries with scores below 6.0 are allowed to be executed. A Naive Bayes classifier was able to correctly classify 142 of 239 queries from the experiment presented in the paper.

In [51], an exploration into a machine learning approach for SQL query response time estimation in cloud services was presented. The premise of the work is that most cloud applications are data-driven and the DBMS supporting these cloud applications are critical in the cloud software ensemble. As a result, this work emphasized the importance of maintaining the quality of service (QoS) in the cloud applications with respect to the time and resources required to execute SQL queries on the DBMS implemented as part of the cloud services. The paper states that awareness of the execution time of an incoming query is a crucial factor in the decision-making process of cloud systems. The machine learning modelling of SQL query execution time in the cloud was approached using regression algorithms. The methodology employed by the work in [51] represented words in their SQL query samples as tokens and used Bag-of-Words (BoW) technique to extract the features of their dataset. Similarly, the BoW approach is also used in this research work to process and represent the SQL dataset for the machine learning experiment, as presented in chapter 5. For the regression problem, the paper presented three algorithms: Gradient Boosting Regression (GBR), Random Forest Regression (RFR) and Support Vector Regression (SVR). It was reported that these were the best-performing algorithms for their dataset.

Another similarity in the approach used in [51] and this research is the method for configuring hyperparameters for machine learning models. Machine learning model training requires hyperparameter tuning, which is typically specified using heuristics. This is because machine learning models have parameters that are not directly derivable from the dataset and there are no analytical formulas available to calculate an appropriate value [90], learning the value of

these parameters is subject to tuning (hyperparameters). The paper implemented a grid search technique to determine parameter configurations, thereby improving model accuracy. The same technique is used in this research work to enhance model performance. The paper concluded by stating that their approach was able to develop a model that uses dataset features to predict the response time of a database, with acceptable error.

Work	Aim	Dataset	Algorithm(s) Implemented	Metrics
Kamra et al. [83]	Anomalous access pattern recognition	SQL query logs	Naive Bayes	False Positive and False Negative Rates
Jayaprakash, et al. [78]	Automated IDS	SQL query logs	Naive Bayes	Threshold scores
Farias et al. [51]	Query response time estimation	SQL queries	Gradient Boosting Regression, Random Forest Regression and Support Vector Regression	Mean Absolute Error
This research	Data re-identification attack attempt recognition	SQL query logs	Multilayer Perceptron, Naive Bayes, K-Nearest Neighbors and Logistic Regression	Accuracy, Precision, Recall, F-score, AUC, False Positive Rates

Table 3.1: A Comparison Review of Work in Database Query Recognition

The amount of work exploring the relationship between database queries and the application of data mining techniques is relatively limited. Based on the work done in reviewing literature about database queries and data mining, this research is the first attempt at recognising SQL query patterns with re-identification attack tendencies and applying data mining techniques to classify these query patterns. Table 3.1 depicts a comparison review of related work in database query recognition.

3.5 Data Mining and Re-identification Query Pattern Recognition

This research aims to recognise re-identification attacks automatically by studying the pattern in the SQL queries being executed on an anonymised database. To achieve this, machine learning algorithms are employed. The learning capability that can be accomplished through machine learning techniques can be supervised, semi-supervised or unsupervised [119, 8]. Since SQL is a standardised database query language with a set syntax, a distinction can be made to a certain degree between the patterns of *normal* and *abnormal* database queries. However, this may vary from one database to another. This work is focused on solving the problem of recognising re-identification queries (abnormal), and to achieve this, machine learning algorithms are trained based on query patterns re-identification attack attempts are perceived to be. As presented in section 4.6, SQL query data is gathered for both *normal* (non re-identification queries) and *abnormal* (re-identification attempt) query patterns. This allows for explicit labelling of the research dataset.

The parallel between identifying re-identification attack attempts and machine learning is that the system that this work is aiming to develop will function based on its ability to find similarities between user query patterns that are attempting a re-identification attack. Similar to the re-identification experiment analysis and results presented in section 4.4, the system will be making its decision based on the level of probability that any particular user query pattern is re-identification or not. There are machine learning classification algorithms suitable for this type of scenario [8], which is why the research problem to automatically and pre-emptively identify re-identification attacks on anonymised databases is approached as a classification task.

3.6 Supervised, Semi-supervised and Unsupervised Learning

Supervised data mining or supervised learning depends on training data samples from a dataset where labels had already been assigned. This learning technique works by having a teacher or supervisor categorise the training data into classes. In supervised learning, after the training of the model has been achieved, the model begins to generate predictions based on its training and improves itself when new knowledge is acquired [119, 111]. Supervised learning is applicable when the dataset is gathered in a scenario whereby the dataset can be categorised (labelled). This suggests that the researcher is aware of what features the different datasets gathered for experimentation possess. In this research, the dataset was gathered such that it was possible to label *normal* query patterns and *abnormal* query patterns. Supervised learning is chosen for this research experiment because this work is mostly concerned with what features are contained in re-identification attack attempts. Therefore, the *abnormal* class is the target class in this.

Supervised learning works by using the training data to teach the algorithm to produce the desired output. The supervised learning method can be divided into two types of problems when learning. These problems are classification and regression [68]. This research work uses labelled data because data generated from the data collection experiment are distinguishable, the data collection for this work is detailed in 4.6. This distinction allows for the data to be organised under two labels. The first label consists of SQL query data while the participants of the experiment are innocently and curiously interacting with the synthetic dataset while the second label consists of query data while participants are attempting to perform re-identification on the anonymised synthetic dataset, with the two data labels being re-identification attempt data and everything else counting as normal.

Semi-supervised learning operates by making use of both unlabelled and labelled datasets. The majority of datasets used in the model training for semi-supervised learning are unlabelled. It is the balance between supervised learning (fully labelled dataset) and unsupervised learning (fully unlabelled dataset), and it is applicable when only a portion of the dataset under study is labelled. Gathering a labelled dataset is typically time-consuming, and requires human effort to assign labels to samples from the dataset [176, 6]. In unsupervised learning, the dataset is not labelled and the model has to learn through studying and discovering structures in the data. Clusters of data with similar patterns are created, and then the models look for relationships among the created clusters [111]. Unsupervised algorithms find hidden patterns and similarities in the dataset without the need for a human analyst to assign labels [45].

The data points in a dataset are also referred to as *features*, hence the supervised, semi-supervised and unsupervised learning methods are also known as *feature selection* methods [176].

3.7 Classification Problem in Re-identification Query Pattern Recognition

The aim of this research work is a classification problem. This is because the classification method uses an algorithm to accurately assign data into labelled categories that serve as examples to develop a model [15, 68]. It is a predictive modelling problem whereby the label for a class is predicted for an instance of the input data. The labelling is done by observing the attributes of the data objects and grouping these objects based on similarities in their attributes. The labelling of the data in this work assigns data into a category based on whether they contain re-identification

attributes or not. This was determined based on the research intent to recognise patterns that constitute re-identification in database queries. A classifier model performing accurate *generalisation* of the new unseen dataset is the ultimate goal of classification tasks. Generalisation is the ability of the classification model to recognise the underlying features and relationships between attributes of the training dataset, as opposed to a model that exactly matches the training data (overfitting) or one that fails to learn and represent significant underlying relationships among features in the training dataset (underfitting). Both overfitting and underfitting produce a poor-performing classifier that is incapable of accurately classifying new unseen data samples (*generalise*). A classifier that can not generalise is not practicable in real-world domains, thereby defeating the purpose of the classification task [91, 21].

The conventional method of classification uses all the classes in the training data to build models, these models are taught to discriminate between classes. *Discriminatory* methods of classification are suitable for data mining scenarios where there is the availability of data from all classes to build a discriminatory function that distinguishes between the different features of the classes [12]. There are different types of classification problems, each of which is suitable for different training data scenarios. The different types of classification tasks are *one-class*, *binary* and *multi-class* classification.

One-class Classification

The method uses data from a single class (target class) to build a model and functions based on its ability to recognise data from that particular class. Data from the other class (outlier class) is rejected. The aim here is to recognise instances of an idea by only training the model with examples from that idea [114]. The one-class method becomes the method of choice when there is an overwhelming amount of data for a single class over the other class, or the other class is non-existent (imbalanced data) [12]. This work does not have a case of imbalanced data, equal samples of dataset was gathered for the two classes of experiment data.

Binary Classification

This is regarded to be the simplest type of classification, also known as two-class classification [85]. The simplicity is due to the amount of effort used in the data collection procedure. For binary classification, the dataset must be gathered in a format whereby the class each of the data sample belongs to is predetermined. Having the dataset in a format that makes it easy to label should be integrated in the data collection process. In a binary classification problem, the training data is classified into two discrete categories. This is suitable for a problem with only two possible outputs. Binary classification is the best fitting classification method for this research model to predict if a series of SQL queries on an anonymised is attempting re-identification or not. The classification problem in this research informed the research methods and data collection strategy presented in chapter 4.

This research is treating the data mining analysis of its dataset as a binary classification problem because it is only concerned with recognising if a series of database query is a re-identification attempt or not. This implies that the problem can be answered with a boolean output (0 or 1 , *yes* or *no*). The aim of this work is developing a system that can identify re-identification query patterns and predict the likelihood of a set of user queries being re-identification (1 , *yes*) or not re-identification (0 , *no*). Any query pattern that does not fit the re-identification query pattern

is regarded by the system to be in the *normal* class. This is more accurate and productive for achieving the research aim. Due to the nature of the data mining task in this work, adopting a binary classification method fits into the research methodology and therefore informs the data collection strategy detailed in section 4.6, as the data collected are categorised into two specific classes; re-identification queries (*abnormal*) and non re-identification queries (*normal*). With this, the research develops a classifier that will study a new, unseen user database queries and predict the probability that the query is attempting re-identification against any particular database or not. The model does this by outputting a value between 0 and 1 . The closer the output value is to 0 or 1 , the lower or higher the likelihood that a series of input query belongs to the re-identification class or not.

Multi-class Classification

A multi-class classification problem aims to classify datasets with three or more class labels. Training data instances are classified into one of the several known classes [4]. Due to the nature of the data mining task in this work, there are only two labels available to be assigned: which are re-identification attempt queries and non re-identification attempt queries. The research model does not fit into the multi-class classification problem.

3.8 Application of Classification Algorithms in Re-identification Query Pattern Recognition

Classification algorithms are widely implemented in the development of data mining models to extract meaningful knowledge from large amounts of datasets. There are different types of classification algorithms and they can be evaluated and selected for any particular classification problem based on several criteria, including accuracy, computational complexity, speed, robustness and scalability [39]. To implement a pattern recognition method that is capable of making a distinction between re-identification attempt queries and non-re identification queries, the pattern categories must be established (supervised learning). Applying classification algorithms in re-identification query pattern recognition involves data collection and preprocessing, tokenization (feature extraction), classification algorithms training and testing [138]. This process is presented in detail in chapter 5 of this work. This work explores the implementation of binary classification algorithms to develop a model that can recognise and predict re-identification patterns in a series of SQL database queries. The classification experiment for this work utilises the functions of built-it programming libraries in Python programming language to implement different classification algorithms. The machine learning algorithms implemented in this work are selected based on their use in binary classification and database query pattern recognition.

3.8.1 Neural Network

Neural Network (NN) or Artificial Neural Network (ANN) is a supervised machine learning algorithm that operates by attempting to recognise fundamental correlations in datasets. Neural networks simulate the operation of a human brain (biological neural network) to make relevant connections in a large amount of input data points. To generate the desired output, neural networks adopt a series of algorithms in the development of its model. A Neural network is typically

comprised of three (3) layers; the *input layer*, a *hidden layer* and an *output layer*. The input layer is responsible for feeding data into the neural network during training, the hidden layer performs the data processing with the use of specified algorithms and the output layer takes the processed information and makes a decision that generates an output value. Neurons at the hidden layer are responsible for processing the input data and generate a desired output. The hidden layer processes implement algorithms known as *activation functions* [158, 2].

The implementation of a neural network can be executed using different architectures. The type of NN architecture employed for this research is known as a *feedforward neural network* (FNN), as depicted in Figure 3.1. A feedforward neural network architecture is also known as a *Multilayer Perceptron (MLP)*, as it consists of an input layer, one or more hidden layers and an output layer [44]. MLPs have been vastly implemented in a range of machine learning problems including pattern recognition, intrusion detection systems, time series forecasting and dynamic modelling [169, 140]. A feed-forward neural network propagates (“feeds”) the output from a layer to the next layer in a forward direction (from input to output layer) until the output value is generated at the output neuron. The connections between neurons are always directed from lower layers to upper layers (from input to output) [121, 147]. Other types of neural network architecture are Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN). CNNs are recognised for their high performance with image and audio input data while RNNs are widely employed in music generation, sentiment analysis, and machine translation applications [46, 47].

In a feedforward neural network architecture, when every neuron in a layer is connected to all the neurons from the proceeding layer, it is referred to as a *dense layer*. This implies that each neuron in the dense layer receives input from all the neurons in the previous layer (fully connected). During the Python implementation of a neural network model for this research, as detailed in section 5.3.1, dense layers are used in the definition of the model. Dense layers have trainable parameters such as activation functions. Deeply connected layers for an MLP are created using *dense layer* from *keras* library in Python [21].

Weight and Bias

Building a neural network requires creating layers of neurons. A neural network layer consists of neurons (also referred to as nodes or units), where a series of computations occur and the product at one layer gets transferred to the next layer for further computations until the output layer, where the desired value is generated. At each neuron, the input data values are multiplied with a set of *weights* to generate the output value at that neuron, to be transferred to the next layer. This calculation is executed using matrix multiplication. The sum of the input multiplied by weight calculations at each node is then transferred through an activation function to generate the output (activation) of each node. This value is referred to as a *summed activation of the node* [21]. *Weights* are the indication of how significant an input data is, based on its features. It signifies the strength of the connection between neurons and it is a factor in the level of influence a change in input has on the output [14].

Besides the input data and weight, another essential parameter in the training of a feedforward neural network is *bias*. Bias in FNNs has the effect of increasing or decreasing (shifting) the net input of the activation function, depending on whether it is positive or negative, respectively [171]. When implementing a FNN, the bias input value is traditionally set to 1 [21, 56]. Bias is a constant value added to the product of input values and weights. A bias value is used at the

input and hidden layer of the neural network. A bias term can be represented as a *bias neuron* in feedforward neural networks, as shown in Figure 3.1 with the b_1 and b_2 neurons.

Backpropagation

The algorithm used by a feedforward neural network to train and learn from a set of data is known as *backpropagation*. The FNN learns by recursively processing a set of training data and checking the neural network model prediction value for each input data against the actual value. The backpropagation algorithm operates by setting a random value for the weights of the input data in the training sample (*weight initialization*). These weight values are then modified (updated) to reduce error or loss in the neural network system. The updates and modifications in the weights are propagated “backward” from the output layer to the hidden layer. Algorithm 1 below presents the steps in the operation of the backpropagation algorithm used in feedforward neural networks [160, 131, 157, 21, 14].

Algorithm 1 Backpropagation Algorithm

1. Initialize the weights and bias in the neural network
 2. Feed the training data as inputs to the network
 3. Propagate the inputs forward, applying activation functions to generate output at each neuron
 4. Backpropagate the error computed at the output layer back to the hidden layer
 5. Modify and update weight values to reflect the backpropagated errors
 6. Repeat until a predetermined iteration cycle is complete, then terminate
-

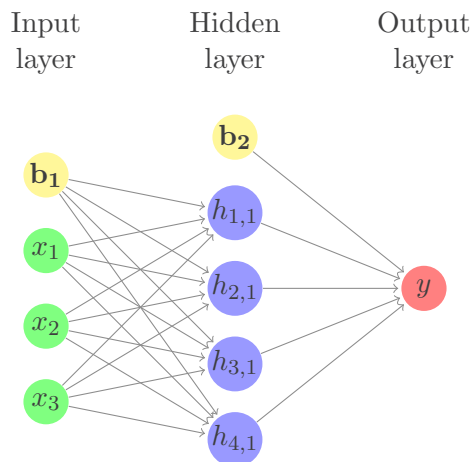


Figure 3.1: Architecture of a Feedforward Neural Network

Activation Functions

An activation function in neural networks controls the processing of the summed weighted inputs from the neurons to generate an output for a particular input. The activation function is implemented in the computation of the output value of a neuron to delimit the output value within a controlled range. The output value from each neuron is boundless and may produce any value from negative to positive infinity. Activation functions are used to solve the problem of boundless outputs, as an activation function limits the range of the output of a neuron to a ranged value [171, 14]. The activation functions used in training a NN for this research are the *sigmoid* function and the *Rectified Linear Unit (ReLU)* activation function.

- **Sigmoid Function:** The sigmoid activation function is a nonlinear function that assumes a continuous range of values from 0 to 1. It is implemented in the output layer of a neural network for binary classification, due to the range (boundary) of its output values. Values of input that are above the boundary of 1 are transformed back to 1 and those below 0 are changed to 0. The function is centered at 0.5 and performs classification of an output based on how close to the threshold such output is. In binary classification, the threshold is a value between 0 and 1 that denotes class membership for a data sample based on the prediction value. 0 is the *normal* class and 1 is for the *abnormal* class [93, 61]. A nonlinear function is used for the activation in the nodes for this research's neural network because a nonlinear activation function (sigmoid) allows the neurons to learn complex patterns in the data.

A drawback in the performance of sigmoid activation is its tendency to *saturate*, meaning larger input values snap to 1 and smaller values to 0. Since the function treats all possible inputs to be from 0 to 0.5 and 0.5 to 1, sigmoid is highly sensitive to changes around the midpoint (0.5). Once saturation happens in the function, it becomes difficult for the algorithm to adapt to weights and make a good judgment about how relevant the information contained in an input data is to training the model. This problem in the sigmoid function is known as *vanishing gradient* [21].

- **Rectified Linear Unit (ReLU) Function:** To address the vanishing gradient problem in the sigmoid activation, Rectified Linear function became the standard. A unit or node that

applies the function is known as Rectified Linear Unit (ReLU) and it prevents saturation of input values in the neural network. Replacement of the sigmoid function with ReLU at the hidden layer of a neural network has largely contributed to improvement in the performance of neural network models [58, 21]. This research is implementing the ReLU in its neural network to avoid vanishing gradients, as ReLU is not subject to this problem. ReLU allows a gradual change in the output value. The desired output for this work is to determine the probability of a set of queries belonging to one of two classes, therefore, the ReLU function is used in the hidden layer of the neural network design for this work to avoid the vanishing gradient problem.

There are other activation functions used in the building of a neural network, these include hyperbolic tangent (tanh) and softmax activation functions. The tanh function has a range of -1 to 1, with a center of 0, and is mostly applicable when the desired output is not 0 and 1. The softmax activation function is applied in the implementation of multi-class classifiers. It combines multiple sigmoid functions and returns the probability of all individual classes in the classification task. [170, 139].

Loss Functions and Optimization

Neural networks are required to have a loss function when they are being configured and trained. A loss function also referred to as an *objective function*, *cost function* or an *error function* measures the deviation in the neural network model's ability to map a predicted value with an actual value (*error or loss*). The loss function works to minimize the error in the neural network. There are different types of loss functions, and a loss function that represents the aim of the research classification problem must be selected for the neural network design in this work. *Cross-entropy* and *Mean Squared Error* are two broad types of loss functions. Practically, cross-entropy loss functions are implemented in classification problems and mean-squared error loss function are utilised in regression problems. Neural networks are trained using cross-entropy as the loss function. A NN model designed for classification with a sigmoid activation function at the output layer learns quicker and more efficiently when implementing a cross-entropy loss function [21, 126]. Cross-entropy computes a value that summarizes the average difference between the predicted and actual value for the target class. *Binary cross-entropy* loss function is used for the neural network training in this work.

The optimization algorithm implemented in neural networks is broadly known as *gradient descent*. Specifically, the *Mini-batch gradient descent* is used while training the neural network for this research. The algorithm conducts a training iteration for a specified n amount of training data sample (*batch size*) and then updates the weight parameter for every mini-batch of n training samples. A calculation of the error for the model's current state is done and backpropagation is used to update the weights of the model. This process is known as *optimization*. The neural network model will continue to modify its weight parameter until the lowest possible loss is achieved [21, 129, 48]. There are three different types of gradient descent algorithms, including mini-batch gradient descent. Others are Batch gradient descent, where the error is computed for the entire data sample before one weight parameter update (batch = size of the training set), and Stochastic Gradient Descent (SGD) which updates weight update for each training sample one at a time (batch size = 1). Mini-batch gradient descent combines the advantages of the other two types to provide a more stable and efficient optimization algorithm. It creates a balance between the computational efficiency provided by batch gradient and the speed of SGD [48]. Due

to its advantages, mini-batch gradient descent is the algorithm to be used when training a neural network model [129].

During optimization, weights get updated as errors are backpropagated. The amount of weights that gets updated is known as the *learning rate* (also referred to as step size). The neural network's learning rate manages the speed with which the model learns. A high step size increases the speed of the neural network's learning, however, this can lead to the model training concluding with a poor set of weights. On the other hand, a low step size allows the model to learn optimally but the training period becomes remarkably longer. Both of these scenarios lead to high training errors. To resolve this problem and create a balance in the learning rate of the model, an *adaptive learning rate* method is implemented to monitor the performance of the model on the training dataset to adjust the learning rate. There are three types of adaptive learning rate namely *Adaptive Gradient Algorithm* (AdaGrad), *Root Mean Squared Propagation* (RMSProp) and *Adaptive Moment Estimation* (Adam) [21, 126]. Due to its computational efficiency and minimal memory requirements [87], the Adam optimizer is used in this work to adapt and maintain the learning rate of the NN model during training.

3.8.2 Naive Bayes

Naive Bayes (NB) is a collection of probabilistic classification algorithms. The algorithms can be implemented in binary classification problems to develop classifiers. Naive Bayes classification works based on a “naive” assumption that all features in a data sample are independent of each other given the class value [166]. Naive Bayes algorithms are supervised learning methods that have their foundation on Bayes' theorem. The Naive Bayes algorithm estimates the probability of an observation (dataset features), given a class. For instance, the algorithm makes an initial assumption that a series of SQL query patterns belong to the re-identification attempt class, then it estimates the probability of observing that particular SQL query pattern as a re-identification attempt given that it belongs to the re-identification class. It estimates the probability for the existence of features in a sample data, given the frequency of occurrence of that feature in a dataset category (class) based on the training data [173]. This assumption made by Naive Bayes is known as *class conditional independence* [138]. The different naive Bayes classifier algorithms have their differences predominantly in the assumptions they make about the probability distribution [135]. There are different Naive Bayes algorithms implemented in the scikit-learn python library, including Gaussian Naive Bayes (GNB), Bernoulli Naive Bayes (BNB), and Categorical Naive Bayes (CNB) [135]. The Multinomial Naive Bayes (MNB) classifier is suitable for the classification of datasets with features such as word counts in text data classification [136]. The essential hyperparameter to be tuned in the implementation of multinomial NB is *additive smoothing* (*alpha*) value. The value is set to 1 by default in scikit-learn and a value of 0 is used to have no smoothing in the algorithm. Smoothing is a method for making better estimation of probabilities of dataset features when there is insufficient data available to calculate accurate probabilities in the dataset. In additive smoothing, the frequency of occurrence of a data feature is increased by 1, then the algorithm calculates the probability of each feature given a class [159]. The implementation of Naive Bayes classification in this work is presented in 5.3.2.

Naive Bayes classifiers are known to be simple models [79, 166], because of the “naive” assumption it makes is inaccurate in a majority of real-world tasks and scenarios. However, NB classifiers have been satisfactorily applicable in many real-world domains for document classification, and email spam filtering [101, 102]. Naive Bayes has been implemented in text classification, cancer classification, and fake news detection [123, 82, 106].

3.8.3 K-Nearest Neighbours

The K-Nearest Neighbors (KNN) algorithm is a supervised classification method that predicts the class membership of a testing data sample based on the closest (nearest) neighboring training data features [150]. KNN is an example-based (instance-based) learning or lazy learning algorithm, this is because KNN simply stores all the training instances from the training dataset and any computation to be done is postponed until the algorithm is tasked with the classification of a new data sample [138, 35]. When new input data is introduced to the algorithm, the k nearest neighbors for the input are estimated and the class label with the majority of neighboring data points claims the classification of the new input for its class. The class with the most frequently represented training samples among k nearest training samples claim a new test sample data to the class [84]. Due to the k nearest neighbors estimation in the KNN classifier prediction decision-making, the value of k is a significant hyperparameter required to be optimally tuned when training a KNN model [150]. The value is an odd number, to avoid having a tie between classes. A large k value reduces the chances of overfitting in the model, however, underfitting tends to occur if the value is too large. [150, 174].

The presupposition of k-Nearest Neighbors is that the classification of an unseen instance (unknown) can be achieved by relating the unknown to the known using a distance function (similarity function). Besides the value of k , another hyperparameter that is essential to the KNN model development is the *distance function* to be implemented [174, 84, 86]. The distance function is used to estimate the instances from the stored training dataset nearest to a new unseen instance. The function locates the nearest training instances and predicts the label for the new instance's classification. Scikit-learn implements the Minkowski (L_p) distance metric. Minkowski distance is a generalised metric that includes a family of different distance metrics that are specific cases of the Minkowski metric. The distance metrics in the Minkowski family include Euclidean (L_1) and Manhattan (L_2) distance metrics. The choice of the metric to be used is declared by specifying the value of p . p represents the power parameter for the Minkowski metric [132]. When $p = 1$, Manhattan distance is used and $p = 2$ uses Euclidean distance [116]. There are different distance functions implemented in the Python scikit-learn library, however, most instance-based learning algorithms use the *Euclidean function* [164]. To measure the similarity (distance) between new SQL query instance and the training dataset, this research explores the Euclidean and Manhattan distance functions in the KNN implementation for the classification of re-identification attack attempt queries. This implementation is presented in 5.3.3.

KNN is known to be effective in several applications [86], however, a notable disadvantage of the KNN classifier is its inefficiency with large datasets, as this will require a significant memory usage to store all the training data instance due to KNN's lazy learning. A KNN model is also subjected to a high computational cost at the point of classification, as all computation is done by the model at this point [84].

3.8.4 Logistic Regression

Logistic regression (LR) is a statistical model that estimates the probability of an instance being true or false [69], for example, if a SQL query sample is a re-identification attempt or not. It is a widely applicable machine learning algorithm for data mining scenario that involves binary classification and problems that requires a prediction-based model. Logistic regression has its application predominantly in binary classification. A classifier that uses logistic regression takes sets of inputs and uses a function to determine the output, that is the classification of the input

data [177]. Logistic regression is a probabilistic algorithm, the output of logistic regression from an input dataset is the probability that the input data is negative or positive (0 or 1, yes or no) [53]. Due to this built-in operation of the algorithm, it is widely applicable for binary classification problems. More importantly, it is very relevant and implementable in the process of achieving the aim of this research's machine learning objective. This research implements logistic regression to train the research dataset. Logistic regression also has application in statistical pattern recognition [41, 31]. Logistic regression classifiers use the sigmoid function for their operation. The sigmoid function is also referred to a *logistic function* [41] and the probability expressing the possible outcomes of a test sample is modeled using this function [134].

The implementation of logistic regression in Python scikit-learn library utilises *solver* algorithms for optimization (parameter update) in the model during training to reduce errors in the model. This is an essential hyperparameter to tune when training a logistic regression model. The different solvers available in scikit-learn are applied in the experiment presented within this work. Logistic regression in scikit-learn implements regularization to improve the model's performance and prevent it from overfitting the dataset. *Regularization* limits the value of parameters towards 0, thereby reducing the chances of creating a complex model that overfits that dataset [14]. The implementation of logistic regression in scikit-learn applies regularization by default [133]. The *C* hyperparameter is another essential value to be tuned to improve model performance. The hyperparameter is referred to as the *inverse of regularization strength*. The value of *C* must be a positive float, and smaller values of *C* provides stronger regularization [133]. A low value of *C* controls the parameters in the model, to avoid excessive complexity and eventual overfitting. There are five solvers available in scikit-learn for logistic regression optimization. They are known as *newton-cg* (Newton Newton Conjugate Gradient Method), *lbfgs* (Limited-memory Broyden-Fletcher-Goldfarb-Shanno), *liblinear* (Large Linear Classification), *sag* (Stochastic Average Gradient descent) and *saga* - an extension of the *sag* solver.

3.9 Classification Performance Evaluation Metrics

When building a model for classification, an essential criterion in the assessment of the quality of the model's training (learning) and testing is the performance evaluation metrics employed. Evaluation metrics in the training phase optimize the classification algorithm, to discriminate and adopt the best solution and produce an accurate prediction. There are several existing metrics that can be applied, depending on the classification scenario. However, these performance evaluation metrics in machine learning classification problems are broadly categorised into three different types. These categories are: Threshold-based metrics, Rank-based metrics, and Probabilistic metrics [54, 65, 167].

Threshold-based performance metrics are used when the classifier is required to operate with a minimal level of errors. Metrics based on thresholds retain a high level of human relatability, therefore are employed in many real-world applications of classification models. Threshold-based metrics include *accuracy*, *precision*, *F-score* and *recall* [167]. Rank-based metrics assess the overall ranking performance of the classifier. The Area Under the ROC (Receiver Operating Characteristics) Curve (AUC) is a widely used ranking type metric [65]. Probability metrics are based on the probabilistic understanding of error, it involves assessing the deviation of an output from true probability. Probability-based metrics include Mean Absolute Error (MAE), Mean Squared Error (MSE), and LogLoss (Cross-entropy). These metrics assess the reliability of classifiers, measuring when the classifier fails and when it predicts a wrong class [54].

In binary classification, the threshold-based performance metrics are adopted and these metrics are defined using a *confusion matrix* [65, 113]. Studies [122, 64, 24, 60, 156] have either shown or stated that good accuracy is the primary concern and most practicable evaluation metric in binary classification problems. With the accuracy metric, the quality of the classifier model is evaluated based on the percentage of correct predictions over the overall instances. Another metric that complements the accuracy metric is *error*. The error metric assesses the classifier model using the percentage of incorrect predictions. A confusion matrix is a tool used for the analysis of how effective a classifier is at recognising data records of different classes. A confusion matrix is a tabular visualisation of the classifier’s performance evaluation. The columns of the table show the instances of a predicted class and the rows are for instances of an actual class. A confusion matrix has a $n \times n$ size, in relation to the classifier and presents the predicted and the actual classification of the dataset. Confusion matrix is also referred to as a *contingency table* [156, 113, 151]. The classification problem in this research work is binary, therefore $n = 2$. The confusion matrix for the classification of SQL queries as either a re-identification attack attempt or not is represented in Table 3.2.

	Predicted Negative Class	Predicted Positive Class
Actual Negative Class	true negative (tn)	false positive (fp)
Actual Positive Class	false negative (fn)	true positive (tp)

Table 3.2: Confusion Matrix for Binary Classification

The true positives (tp) indicates re-identification attempts that are correctly classified as re-identification attempts by the model, while true negative (tn) show non re-identification attempts classified as such. False positives (fp) and false negatives (fn) on the other hand represent non re-identification attempts inaccurately classified and non re-identification attempt queries accurately classified respectively.

Accuracy, Precision, Recall, and F-Score

Achieving the data mining aim of this research has been set to be a binary classification problem, in which the classifier predicts the likelihood of a set of SQL query data being a re-identification attack attempt or not. Binary classification evaluation metrics are used, as these are empirical measures to ascertain how effective the classifier model is at correctly predicting a re-identification attack attempt. These metrics are *accuracy*, *precision*, *F-score* and *recall* [151, 144], they are defined and calculated as follows:

- **Accuracy:** This is the most prominently used criterion to measure the classification performance of a classifier model. It is defined as the ratio of the total number of correct predictions to the total number of predictions, it divides the correctly classified data samples (query patterns) by the overall data samples [63, 144]. It can be expressed as shown in equation 3.1.

$$Accuracy = \frac{tp + tn}{tp + tn + fp + fn} \quad (3.1)$$

A complementary metric of *accuracy* in classification performance evaluation is *error rate* (*ERR*) or *misclassification rate* [65, 151]. It calculates the amount of misclassified data

samples from both positive (re-identification attempt query data) and negative (non re-identification attempt query data) classes, it is represented as:

$$ERR = 1 - Accuracy \quad (3.2)$$

- **Precision:** The precision metric denotes the ratio of positive data samples that are correctly classified to the overall number of positive predicted data samples. Precision is also referred to as *Positive Prediction Value (PPV)* [151]. It is the classifier's ability to not predict a negative class sample as positive. The equation to generate a classifier's precision is represented in equation 3.3.

$$Precision = \frac{tp}{tp + fp} \quad (3.3)$$

- **Recall:** Recall, also referred to as *sensitivity* or *True Positive Rate (TPR)* is the ratio of the positive correctly classified query patterns to the overall number of positive query patterns. It implies the ability of the classification model to find all positive samples in the testing test, as expressed in equation 3.4.

$$Recall = \frac{tp}{tp + fn} \quad (3.4)$$

Inverse recall or *True Negative Rate (TNR)* is the ratio of negative samples correctly predicted to the total number of negative samples. This metric complements recall and it is also referred to as *specificity*. It denotes the negative instances correctly predicted. It is expressed in equation 3.5 below.

$$TNR = \frac{tn}{fp + tn} \quad (3.5)$$

False Positive Rate (FPR), a complementary metric for *TNR* represents the proportion of negative samples incorrectly predicted. Calculated as shown in equation 3.6.

$$FPR = 1 - TNR \quad (3.6)$$

- **F-Score:** This is also called *F-measure* or *F1-score*, it is the harmonic mean of precision and recall. The F-score value is in a range between 0 and 1, and a high F-score value signifies a high performance in the classifier model [151]. It is represented in 3.7.

$$F - Score = \frac{2 \times precision \times recall}{precision + recall} \quad (3.7)$$

Area Under the ROC Curve (AUC)

Area Under the ROC Curve (AUC) is used to rank and compare different classification algorithms, as it indicates the overall performance of a classification model [65]. According to the work done in [66], the AUC metric is a better measure than the accuracy metric in evaluating classifiers. The ROC curve is a two-dimensional graph that plots the *True Positive Rate (TPR)* on the y-axis

against the *False Positive Rate (FPR)* on the x-axis as it relates to a classifier’s performance scores on the testing dataset. The two axes are bounded between 0 and 1. This creates a cost-benefit balance in the classifier performance. Costs are the false positives while benefits are the true positives. The ROC curve does not have a single value to represent its measurement, therefore it is difficult to use ROC curves for ranking and comparing different classifiers. The AUC calculates the area under the ROC curve, a value that is always ranged between 0 and 1. Any practicable classifier should not produce an AUC score lower than 0.5 [19, 151]. The AUC metric is implemented in this research work to rank and compare the performance of different classification algorithms. The AUC value in a binary classification task is calculated as shown in equation 3.8.

The goal of the ROC curve is to make the best possible trade-off between the TPR and FPR. The optimum scenario is achieving a TPR of 1 along the y-axis and an FPR of 0 along the x-axis. The ROC calculation utilizes different thresholds to generate different confusion matrices. The class distribution and the confusion matrix is represented by a point (FPR, TPR) on the ROC curve. At varying thresholds, there is a different value between 0 and 1 for TPR and FPR. These values are used to generate the ROC curve. To rank a model’s performance against other models, AUC is calculated for the model’s ROC curve. The AUC is a single value estimated from an entire ROC curve. AUC is used in this work to compare the performance of the different classification models trained for the recognition of re-identification attack attempt queries, this is presented in section 6.1.

$$AUC = \frac{S_p - n_p(n_p + 1)/2}{n_p n_n} \quad (3.8)$$

S_p is the sum of all positive instances, n_p represents the number of positive samples ranked and n_p and n_n is the number of positive and negative samples respectively [66].

3.9.1 Bias and Variance

As mentioned in 3.7, the ability of a classification model to *generalise* and correctly predict the class membership of new, previously unseen data instances is the ultimate performance evaluation metric to determine the real-world applicability of a classification model. The bias-variance evaluation can be used to improve the performance of the machine learning algorithm. The concept of *bias* and *variance* is similar to that of underfitting and overfitting in machine learning models. A model is considered to have *high bias* when it is too simple and not flexible enough to learn the intricate features and patterns in the training dataset, therefore, producing a *high training error* (underfitting). The model performs poorly at testing with both the testing dataset during performance evaluation and the unseen test data. Conversely, a more complex model (relative to the dataset under study) tends to be too flexible and learns the dataset too well, including the noise present in the data. Such a model is considered to have *high variance* (overfitting). In a high-variance model, there is a large disparity between the training error and the testing error. In other words, a model with high variance performs well on the training data but poorly on unseen test data. [14, 91, 21]. Neither the high bias nor the high variance scenario is desirable in a model. Therefore, training and testing of the different models for this research work are monitored to strike an efficient bias-variance balance and produce a classification model with *good fit* (learning enough intricate relationships in the dataset to generalise well on unseen data instances).

3.9.2 Cross-Validation

Cross-validation (CV) is a data resampling method that is used to evaluate the ability of classifier models to generalise and predict accurately, thereby preventing overfitting. Cross-validation is commonly implemented in model hyperparameter tuning [14, 13]. The training of the classification algorithms in this work involves generating an initial baseline model. The baseline model uses default hyperparameter value for training, cross-validation is then used to calculate optimum hyperparameter values in an attempt to improve the performance of the model. A widely applied method of cross-validation technique is *k-fold cross-validation*. This method divides the training data into k subsets and for each k training iteration, $k-1$ data subsets are used as the model training set while the remaining subset is used for *validation*. A different subset is used for validation per training iteration, with training repeating until all k subsets are used as validation sets [14].

In this work, training NB, KNN, and LR involves the implementation of k -fold cross-validation to explore different values (tune) in the classification models, using *5-fold* cross-validation. Arbitrary tuning of hyperparameters can lead to uncontrolled and counter-intuitive effects in the model operation [51]. Therefore, the hyperparameter values with the best fit on the research training dataset are used to develop *tuned* classifier models in this work, as presented in section 5.3. The scikit-learn library implements a grid search cross-validation function (*GridSearchCV*) for an exhaustive search of the specified hyperparameters, to estimate the optimum hyperparameter values for a best-fit model. In the MLP training, a validation dataset is used to assess how well the model is learning the research training dataset.

3.10 Chapter Summary

This chapter discussed the concepts of data mining and machine learning. The necessity of data mining and the application of machine learning to process and infer knowledge from raw data was emphasized. A conceptual distinction between data mining and machine learning was presented, as these terms are often used interchangeably. An overview of techniques used in data mining was explored, investigating the different applications of these techniques. The use of Python programming language and its data science libraries as the preferred choice for this research to analyse its dataset was also discussed. Python libraries have become a standard tool for analysis in data science and mining [145].

This chapter presents the research's aim as a classification problem, thereby informing the research method and dataset collection presented in chapter 5. Different classification problems are reviewed and justification for this work being a binary classification case is presented. The chapter continued to present the classification learning algorithms relevant for use in this work. Multilayer perceptron, Naive Bayes, K-nearest neighbors, and logistic regression are presented as the most applicable algorithms to be implemented, due to their applications in pattern recognition, binary classification, and ability to implement a probabilistic function on the output value. The probabilistic output is relevant to this work because the models to be developed should have the ability to recognise the likelihood of a SQL query data input being a re-identification attempt.

Chapter 3 concluded by presenting an analysis of the different evaluation methods for assessing the performance of classification models. The confusion matrix and its relevance in the estimation of a classifier's performance is also explored. Using the details of a confusion matrix, the equations

to calculate different relevant classifier performance evaluation metrics are presented. These classification performance evaluation metrics are used to evaluate the models trained for this research to recognise re-identification attack attempt queries, this is presented in section 6.1.

Chapter 4

Data Re-identification Analysis and Case Study Selection

This chapter explores the methodological approach implemented toward fulfilling one of the objectives set for this research work, which is to explore the possibility of representing re-identification attacks as SQL queries on a database. The design of the methods, strategies, and data collection procedures is presented at this stage of the research. In section 4.2, instances of successful re-identification attacks are described and used as case studies for an exploratory study in the research. An experiment to recognise re-identification attacks as SQL queries is presented in section 4.3, which includes the creation of synthetic datasets for two of the case studies described. The results of the experiment and the strategy employed are analysed in sections 4.4 and 4.5 respectively. Finally, the campaign to collect SQL query dataset is detailed in section 4.6.

4.1 Exploratory Study Overview

Due to the lack of success in the endeavour to acquire a real-life anonymised dataset (health data from the NHS), this research used stories from some of the more infamous and successful re-identification attack scenarios as case studies, a reconstruction of what the dataset structure was when it was shared with the public is attempted. This is used to create synthetic datasets for experimentation in this research. The synthetic data creation process imitates the anonymisation techniques employed by the data controllers in each of the scenarios. Even though the shared anonymised datasets from some of these re-identification cases are available and accessible online [81, 80], the real-life secondary databases used for re-identification are not as easily accessible. Also, other initially shared datasets ended up being removed from public domains due to backlash from the privacy damage caused by the re-identification of such datasets. Therefore, synthetic datasets will be generated for both the shared anonymised datasets and the secondary databases used for re-identification.

As stated in the introductory section, a major premise of this project work is the idea that re-identification of anonymised datasets can occur through a series of SQL queries, executed against the database by malicious users attempting to breach the data privacy of the subjects in the dataset. With this mindset, an exploratory study to ascertain that re-identification is achievable on an anonymised database by executing a series of SQL queries is conducted, as presented in section 4.3. A SQL query dataset is then collected, involving individuals attempting

re-identification attacks against the anonymised synthetic database created for this research. The log of SQL queries executed by participants in the data collection is recorded and used in the data mining experiment to train classification algorithms.

This research uses *Microsoft SQL Server Management Studio* (MS SSMS) as the DBMS to store and manage the anonymised synthetic databases. Microsoft SSMS is chosen as the preferred DBMS due to some of its characteristics. MS SSMS has been compared to other tools with similar functionalities in various literature using different criteria such as supported operating systems, supported SQL language, syntax, interface, hardware requirements, and most significantly query execution time [71, 77, 10]. In a detailed Microsoft SQL Server and Oracle DBMS, comparison presented by the work in [71], MS SQL Server is found to be simpler and easier to use. This is due to a more user-friendly syntax of the T-SQL (Transact-SQL) used in the MS SSMS. The participants in the data collection process for this research include beginner, intermediate, and advanced users of SQL, therefore, it is more productive to use MS SSMS to accommodate the variations in the skill level of participants. Also, the results of comparing the execution time of the two DBMS showed that MS SQL Server performs better and executes SQL in a significantly shorter amount of time. This remains true across multiple SQL query keywords, queries from a single table, and queries for merging tables [71]. As a result, this research work is making use of MS SSMS in its data collection campaign.

Users (participants in the research data collection campaign) accessed and interacted with the anonymised database on this platform. An integrated tool in Microsoft SQL Data Management Studio called *SQL Server Profiler* is used for the capture of SQL queries executed by participants to interact with the database. The collected SQL query data is analysed to identify similarities in patterns for non re-identification queries and the queries that are believed to have the intention of performing a re-identification attack. With this pool of SQL queries, classification algorithms are trained to learn the patterns in re-identification attack attempt queries on an anonymised database. Figure 4.1 shows a flowchart representing the methodical approach adopted for the exploratory study conducted in this chapter to analyse and understand re-identification attacks from a technical perspective.

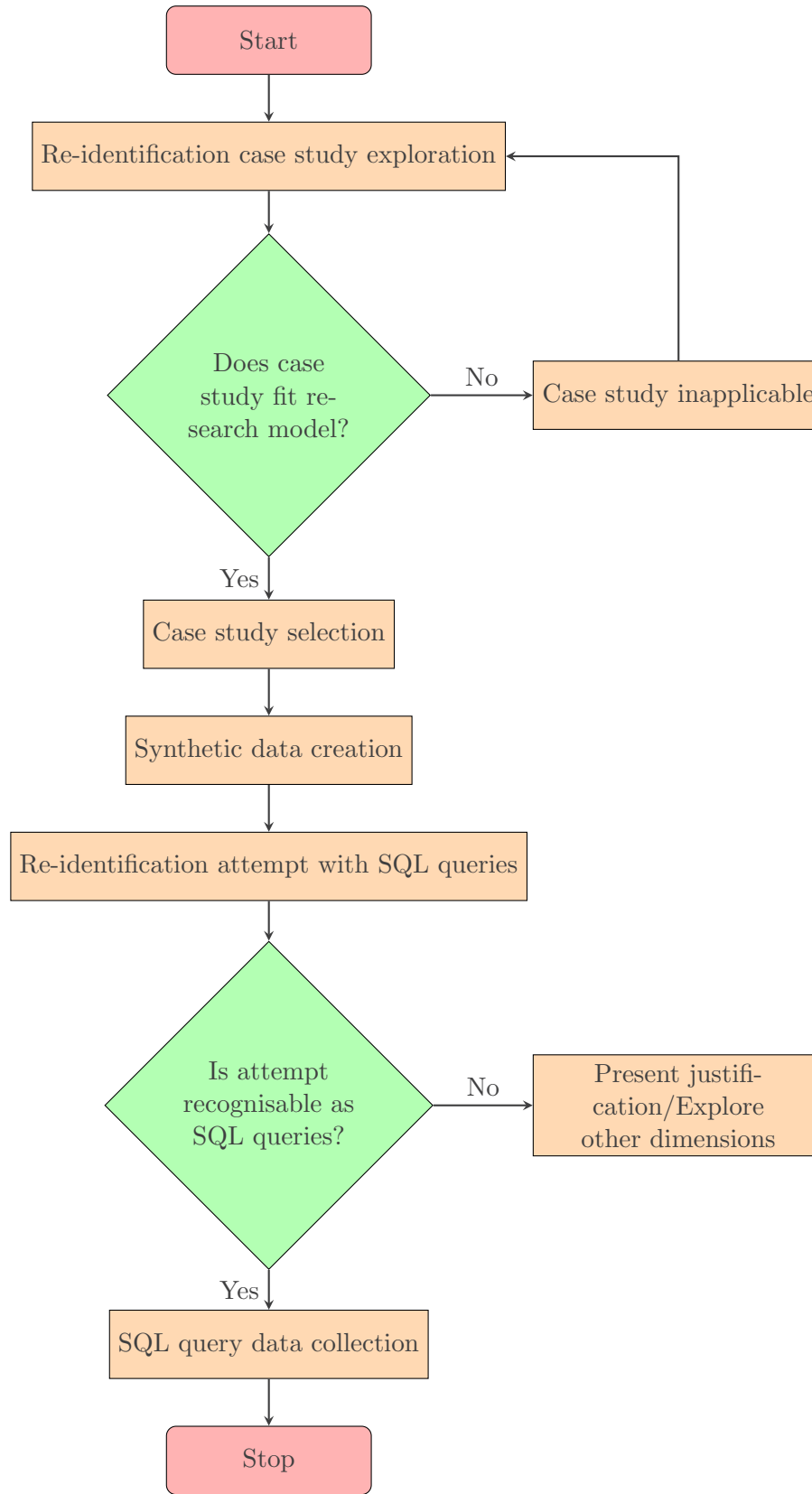


Figure 4.1: Re-identification Analysis and Case Study Selection Approach

4.2 Data Re-identification Case Studies

Over the years, there have been instances of successful re-identification attacks on anonymised databases. This has weakened the level of reliability that can be placed in anonymisation techniques. In each of the re-identification cases, the data controllers believed that the databases were privacy-protected before they were shared publicly. This chapter of the research explores some of the most significant cases of successful data re-identification.

These re-identification attack cases portray remarkable failures of data anonymisation. In every case, the data controller placed unjustifiable trust in their data anonymisation techniques. The case study scenarios demonstrate the prevalent culture of what Ohms in [110] referred to as “release-and-forget” anonymisation (freely sharing anonymised dataset with the belief that it is privacy safe) among data controllers. The case studies also highlight the advancement in re-identification techniques [110].

4.2.1 The AOL Data Release

In 2006, America Online (AOL) publicly shared twenty million search queries, over three months, for 650,000 of AOL’s search engine users. This was done as part of an initiative referred to as “AOL Research” and according to AOL, the aim was to “embrace the vision of an open research community”. The joy felt by researchers who were excited to have such a treasure of data to work with was cut short when both the researchers and the general public realised that the search queries data shared by AOL could be used to identify subject-specific personal data [96, 110].

AOL performed data suppression on obvious subject-identifying details in the search queries dataset, details like the AOL usernames and IP addresses. This was done in an attempt to anonymise the dataset before sharing it with the public. The suppressed details were represented with distinctive identification numbers (pseudonymised), to sustain the dataset’s research usability and allow researchers to correlate queries to users.

Soon after the data release, bloggers and reporters sorted through the data with the aim of either identifying users or rummaging for embarrassing or shocking search queries. In the aftermath of the breach, a lot of embarrassing queries were associated with the identification numbers. The breach then escalated when hints to the identity of one of the data subjects were discovered in the search query data, and reporters were able to associate these queries with the data subject (Thelma Arnold, a 62-year-old woman from Lilburn, Georgia). The backlash of the breach negatively impacted AOL, leading to the company having to fire the staff in charge of the data release [110].

4.2.2 Netflix Prize Dataset

This research employs the Netflix Prize Data (NPD) release as a case study for one of its experiments. In 2006, Netflix published one hundred million data records, divulging how hundreds of thousands of their customers had rated movies from December 1999 to December 2005. In each entry of the published record, Netflix divulged the movie rated, the rating assigned by the customers, and the date the movie was rated. The data record was anonymised by Netflix before being released, anonymisation was done by removing identifying details such as usernames. However, a unique identifier was assigned to each of the users, to preserve rating-to-rating continuity (pseudonymisation). Netflix’s motive for releasing this record was to use the user ratings

to improve their recommendation algorithms, suggesting new movies to customers based on how favourable their ratings are towards similar movies.

Weeks after the data release, Arvind Narayanan and Professor Vitaly Shmatikov [107], researchers from the University of Texas, disclosed that an attacker who is able to gather limited information from another source about a Netflix customer can easily identify such a customer if their record is present in the dataset released by Netflix. Therefore, it is possible to re-identify individuals from the dataset with only a little outside information about their movie preferences. The researcher performed the re-identification by cross-referencing the Netflix dataset with user ratings on the IMDb (Internet Movie Database) website [96, 109].

4.2.3 The Re-identification of Governor William Weld

Latanya Sweeney, a Computer Science graduate student at MIT (Massachusetts Institute of Technology) in the 1990s, Latanya Sweeney, learned that 87.1% of people in the United States of America can be uniquely identified through the combination of their ZIP code, date of birth and gender. She came to this realisation by analysing the 1990 census data [9, 110]. This study changed the misconception that only identity-specific information could reveal details about who an individual is, as it turned out that, even less-specific information can still reveal subjects' identity. 53% of the American people could be identified by their city, date of birth, and gender, and the same could be done for 18% of the population using their county, date of birth, and gender.

To demonstrate the capabilities of re-identification techniques and the negative impact it has on data anonymisation, Latanya Sweeney exhibited a scenario that supports her position. In Massachusetts, Group Insurance Commission (*GIC*) collected and shared patient-specific datasets with up to a hundred categories for over one hundred thousand state employees (that *GIC* acquires insurance for) and their families. *GIC* removed direct identifiers from the dataset, these are details they believe explicitly identify the subjects: like names, addresses, and social security numbers. However, up to a hundred fields of data per subject were still shared, including the significant ones (ZIP code, date of birth, and gender).

Massachusetts state Governor, William Weld had his medical data in the shared dataset. It was already known that the Governor lived in Cambridge, Massachusetts, with forty-four thousand residents and seven ZIP codes. Using the Cambridge voter's list (another publicly available dataset containing the names, addresses, date of birth, gender of every voter in the city), Sweeney was able to re-identify the Governor and linked him to his specific health information from the health data shared by *GIC*. The re-identification was done using 3 distinguishing details (date of birth, gender and ZIP code) about the Governor [9, 110].

4.3 Recognising Re-identification Attacks as SQL Queries

This research is focused on determining the technical constitution of a re-identification attack against a database, by reconstructing the SQL queries that the users of such a database might be executing. Users' access to a database is through SQL queries, so the representation of a re-identification attack process as a series of SQL queries sets a premise for exploring whether or not there is a recognisable pattern in the re-identification attack process of a database. This work employed some of the more famous re-identification attack scenarios to demonstrate possible

reconstruction of a re-identification attack in the form of a series of SQL queries. This section presents an attempt to practically grasp what database re-identification entails, it is an initial step towards establishing whether re-identification attack query patterns have enough in common, that they can be systematically recognised while in progress. The overall aim of this research is to explore whether re-identification attacks can be detected in series of SQL queries. This research work explores one underlying assumption: that re-identification attacks consist of SQL queries.

4.3.1 Experimenting with Re-identification Attacks

This section explores some of the most significant cases of successful data re-identification in the world of data sharing. These scenarios portray a remarkable weakness in data anonymisation measures used in justifying data releases and the impact of re-identification when data controllers place unjustifiable trust in the implemented anonymisation measures. Using the information from the cases outlined in section 4.2, synthetic data to be used in this experiment was generated.

4.3.2 Synthetic Data Creation

The scarcity of details regarding published successful re-identification attacks leaves room for speculation about how a particular re-identification attack could have been achieved. To begin an attempt to practically reverse engineer a re-identification attack scenario, access to technical details of the attack will be imperative. The uncertainty around the idea that re-identification attacks may be reconstructed and understood as a series of SQL queries is the question that this work was focused on exploring. To attempt this, a set of synthetic datasets was generated for these scenarios.

Experimenting With Netflix Prize Data

The Netflix Prize Data was represented by a set of synthetic data that mimics the structure and properties of the dataset published by Netflix. The synthetic data was “anonymised” using the available information about how the Netflix training dataset was scrubbed and presumed to be privacy-safe before being shared publicly [81]. The real dataset about this scenario was not used for the experiment because it would require using a real secondary dataset to complement the published Netflix training data. Although the original Netflix ratings database is still available online [81], the secondary database (IMDb) is not published in a format to be directly applicable for the purpose of this research. The synthetic dataset created for this research work includes 3 tables. A *TrainingData* table with columns for anonymised *User ID*, *Movies*, *Date of Grade* and *Grade*; a *MovieTitles* table including columns for *Title* and the *Year* of the movie; and an *IMDb* table, that server as a secondary data source to employ in the re-identification process. The *IMDb* table has *Username*, *Movie Name*, *Ratings*, and *Date of Rating* columns.

Experimenting With GIC Data

Based on the details gathered about this scenario, the synthetic dataset created included two tables. One for the *GIC* data release and the other for the *Voter Rolls*. The *GIC* table includes columns such as *Patient Name*, *Patient Code*, *Gender*, *Event*, *Event Code*, *Event Date*, *Attending Doctor* and other user-related information. In the *GIC* table, the *Patient Name* column was

suppressed to implement some level of anonymity into this table. The *Voter Rolls* table was represented using columns including: *Name*, *Address*, *Birth Date* and *Sex*.

4.3.3 Re-identification Attack Re-creation

This section of the work provides strategic details on how SQL queries were used to explore the Netflix Prize Data and the Governor William Weld re-identification scenarios. This was done with the aim of re-identification in both datasets. As detailed in section 4.5, the re-identification attack in the NPD scenario is an *existential* attack and the re-identification of Governor William Weld from the GIC dataset is a *target* re-identification attack.

Netflix Prize Data Re-identification

Various SQL queries were executed against the dataset, with the aim of correlating a user to their movie ratings (grading) from the two different databases (the published Netflix *TrainingData* and the public Internet Movies Database, *IMDB* platform).

The attack objective in this case is existential and needs to find a re-identifiable subject in the dataset. In an attempt to approach this systematically, a count function was executed against the *TrainingData* table, using the *User ID* column as the criterion. This is to ascertain the number of rows any specific user ID occupies. The motivation behind this strategy is to check if any of the user IDs stand out more than the others. This would make such a User ID an interesting one to explore. The synthetic dataset showed a somewhat varying output, with a particular User ID with the highest entries in the dataset. This work strategically marked the prominent User ID and other User IDs with high entries noteworthy as the experiment progressed. The result of the query is presented in Figure 4.2.

```
1 SELECT [User ID], count([User ID])
2 AS UserEntries
3 FROM [dbo].[TrainingData]
4 GROUP BY [User ID]
5 ORDER BY count([User ID])
6 DESC;
```

Results		Messages
	User ID	UserEntries
1	user 7	8
2	user 18	6
3	user 5	5
4	user 2	5
5	user 37	5
6	user 1	4
7	user 47	4
8	user 48	3
9	user 49	3
10	user 19	3
11	user 50	3
12	user 6	3
13	user 8	3
14	user 9	3
15	user 10	3
16	user 11	3
17	user 12	3

Query executed successfully.

Figure 4.2: Count function with *User ID* column as criteria

The same strategy was employed using the *Movies* column as a criterion, and the returned table view showed two movies to have 14 entries; the highest number of entries. The two movies are *Black Panther* and *The Expendables*, as in Figure 4.3. The result of this query made two movies the points of interest at this stage.

```

1 SELECT [Movies], count([Movies])
2 AS MovieCount
3 FROM [dbo].[TrainingData]
4 GROUP BY [Movies]
5 ORDER BY count([Movies])
6 DESC;
```

Results		Messages
	Movies	MovieCount
1	Black Panther	14
2	The Expendables	14
3	Source Code	12
4	Black Mirror	12
5	Dexter	12
6	Friends	11
7	How I Met Your Mother	11
8	iZombie	11
9	Spider Man	11
10	Step Up	11
11	The Big Bang Theory	10
12	Chief Daddy	10
13	Avengers	9
14	White Collar	9
15	The Pelican Brief	7
16	White Collar	1
17	The Expendables	1

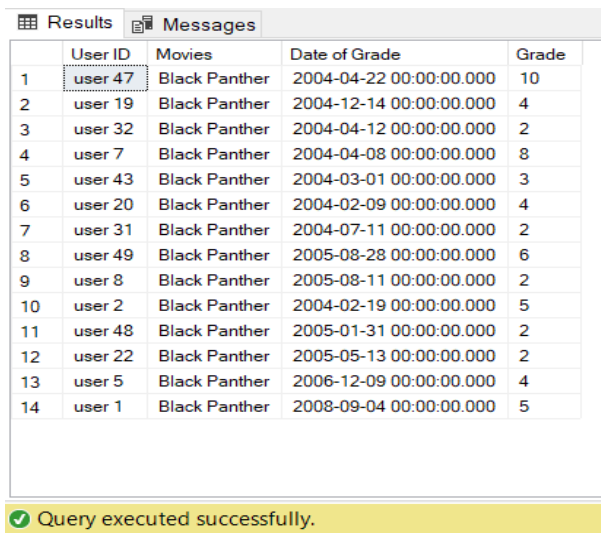
Query executed successfully.

Figure 4.3: Count function with *Movies* column as criteria

After the output shown in Figure 4.3, the next query was targeted towards identifying which 14 users from the *TrainingData* table rated the most recurring movie, using *Black Panther* as the

criterion. The result from the count queries shows that *user 7* has 8 entries and *Black Panther* has 14, making the latter a more prominent feature of the dataset. Therefore, the process proceeded with making *Black Panther* the focus.

```
1 SELECT *
2 FROM [dbo].[TrainingData]
3 WHERE [Movies]='Black Panther'
```



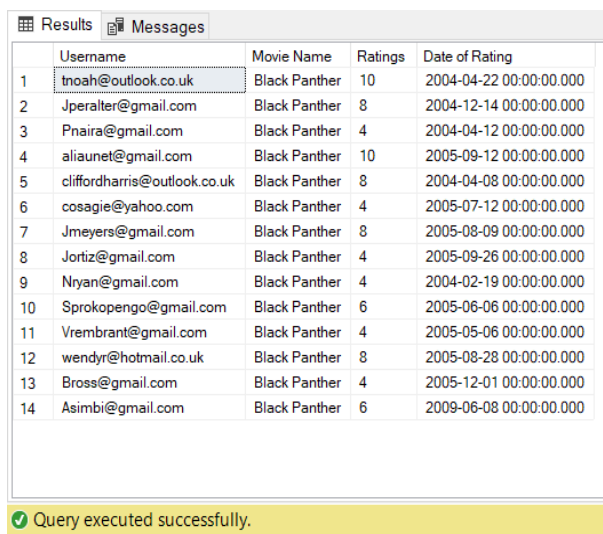
	User ID	Movies	Date of Grade	Grade
1	user 47	Black Panther	2004-04-22 00:00:00.000	10
2	user 19	Black Panther	2004-12-14 00:00:00.000	4
3	user 32	Black Panther	2004-04-12 00:00:00.000	2
4	user 7	Black Panther	2004-04-08 00:00:00.000	8
5	user 43	Black Panther	2004-03-01 00:00:00.000	3
6	user 20	Black Panther	2004-02-09 00:00:00.000	4
7	user 31	Black Panther	2004-07-11 00:00:00.000	2
8	user 49	Black Panther	2005-08-28 00:00:00.000	6
9	user 8	Black Panther	2005-08-11 00:00:00.000	2
10	user 2	Black Panther	2004-02-19 00:00:00.000	5
11	user 48	Black Panther	2005-01-31 00:00:00.000	2
12	user 22	Black Panther	2005-05-13 00:00:00.000	2
13	user 5	Black Panther	2006-12-09 00:00:00.000	4
14	user 1	Black Panther	2008-09-04 00:00:00.000	5

✓ Query executed successfully.

Figure 4.4: Sorting with movie *Black Panther* as criteria

Now that the users that rated “Black Panther” in the *TrainingData*, alongside their rating and its date have been established, the movie *Black Panther* is used as the criterion to sort the users and their ratings in the *IMDb* table. The goal is to compare if there are any similarities in the ratings and the corresponding dates on both tables about the same movie, *Black Panther*. At this point, there is a target criterion in focus, due to the results from preceding queries in the exploration of the *TrainingData*. Using *Black Panther* as criterion, a query to check for its presence in the *IMDb* dataset was executed.

```
1 SELECT *
2 FROM [dbo].[IMDb]
3 WHERE [Movies Titles]='Black Panther '
```



	Username	Movie Name	Ratings	Date of Rating
1	tnoah@outlook.co.uk	Black Panther	10	2004-04-22 00:00:00.000
2	Jperalter@gmail.com	Black Panther	8	2004-12-14 00:00:00.000
3	Pnaira@gmail.com	Black Panther	4	2004-04-12 00:00:00.000
4	aliaunet@gmail.com	Black Panther	10	2005-09-12 00:00:00.000
5	cliffordharris@outlook.co.uk	Black Panther	8	2004-04-08 00:00:00.000
6	cosagie@yahoo.com	Black Panther	4	2005-07-12 00:00:00.000
7	Jmeyers@gmail.com	Black Panther	8	2005-08-09 00:00:00.000
8	Jortiz@gmail.com	Black Panther	4	2005-09-26 00:00:00.000
9	Nryan@gmail.com	Black Panther	4	2004-02-19 00:00:00.000
10	Sprokopengo@gmail.com	Black Panther	6	2005-06-06 00:00:00.000
11	Vrembrant@gmail.com	Black Panther	4	2005-05-06 00:00:00.000
12	wendyr@hotmail.co.uk	Black Panther	8	2005-08-28 00:00:00.000
13	Bross@gmail.com	Black Panther	4	2005-12-01 00:00:00.000
14	Asimbi@gmail.com	Black Panther	6	2009-06-08 00:00:00.000

Query executed successfully.

Figure 4.5: *Black Panther* entries on *IMDb*

The resulting tables in Figure 4.4 and Figure 4.5 were analysed and compared for correlations between the properties (columns) of the table. Section 4.4 presents an analysis of this comparison, from which inferences were made about how the conclusions from the analysis are related to re-identification.

Governor William Weld Re-identification (GIC Data)

Governor William Weld being a public figure, coupled with him experiencing a health episode publicly on live TV made his re-identification from the GIC health insurance dataset a *targeted attack*. There the strategy employed to re-create this attack was to directly hunt for the Governor's information, using the voter list as a secondary dataset: a dataset that contains some of the same information as the GIC data.

Since the Governor's date of birth and the date of his publicised health episode are public information, the re-identification strategy started with exploring the GIC dataset with these already established details. To begin with, a query to count the number of people in the GIC dataset with the Governor's date of birth was executed.

```
1 SELECT *
2 FROM [dbo].[GICdata]
3 WHERE [Date of Birth] = '1945-07-31'
```

Patient Code	Gender	Date of Birth	Zip Code	Event	Event Code	Event Date	Event Time	Hospital	Attending Doctor	Prescription
101	Male	1945-07-31 00:00:00.000	LE2 7GP	Epileptic seizure	7645	2019-01-05 00:00:00.000	1899-12-30 05:32:00.000	Birmingham Children's Hospital	Dr. Who Laure	Paracetamol
109	Female	1945-07-31 00:00:00.000	LE2 1AB	Facial Palsy	9877	2010-09-10 00:00:00.000	1899-12-30 10:30:00.000	Royal Marsden Hospital	Dr. Greg Knox	Muscle Relaxer
102	Female	1945-07-31 00:00:00.000	LE3 4NN	ADHD	2343	2000-06-04 00:00:00.000	1899-12-30 08:10:00.000	Bedford Trust Hospital	Dr. Hall Alley	Vegetables
100	Male	1945-07-31 00:00:00.000	LE1 6TB	Heart attack	754	2019-03-01 00:00:00.000	1899-12-30 14:59:00.000	Freeman Hospital	Dr. House Beckingham	Bed Rest
250	Male	1945-07-31 00:00:00.000	LE3 4NN	Influenza	3476	1996-05-18 00:00:00.000	1899-12-30 10:10:00.000	Deaconess Waltham Hospital	Dr. Nick Jean	Relenza
107	Female	1945-07-31 00:00:00.000	LE2 8YY	Cancer	876	2009-01-20 00:00:00.000	1899-12-30 15:00:00.000	Royal Marsden Hospital	Dr. Coco Oji	Chemotherapy
101	Male	1945-07-31 00:00:00.000	LE2 7GP	Epileptic seizure	564	2019-07-08 00:00:00.000	1899-12-30 05:00:00.000	Birmingham Children's Hospital	Dr. Who Laure	Paracetamol
109	Female	1945-07-31 00:00:00.000	LE2 1AB	Facial Palsy	6544	2009-12-09 00:00:00.000	1899-12-30 09:18:00.000	Royal Marsden Hospital	Dr. Greg Knox	Muscle Relaxer
102	Female	1945-07-31 00:00:00.000	LE3 4NN	ADHD	3454	2020-04-08 00:00:00.000	1899-12-30 06:32:00.000	Bedford Trust Hospital	Dr. Hall Alley	Vegetables
100	Male	1945-07-31 00:00:00.000	LE1 6TB	Heart attack	563	2020-04-08 00:00:00.000	1899-12-30 01:32:00.000	Freeman Hospital	Dr. House Beckingham	Bed Rest
250	Male	1945-07-31 00:00:00.000	LE3 4NN	Diarhea	4556	1996-01-18 00:00:00.000	1899-12-30 12:02:00.000	Deaconess Waltham Hospital	Dr. Nick Jean	Loperamide
107	Female	1945-07-31 00:00:00.000	LE2 8YY	Cancer	7643	2010-03-09 00:00:00.000	1899-12-30 13:59:00.000	Royal Marsden Hospital	Dr. Coco Oji	Chemotherapy
102	Female	1945-07-31 00:00:00.000	LE3 4NN	ADHD	4677	2010-11-10 00:00:00.000	1899-12-30 02:00:00.000	Bedford Trust Hospital	Dr. Hall Alley	Vegetables
101	Male	1945-07-31 00:00:00.000	LE2 7GP	Epileptic seizure	876	2019-12-04 00:00:00.000	1899-12-30 10:15:00.000	Birmingham Children's Hospital	Dr. Who Laure	Paracetamol
109	Female	1945-07-31 00:00:00.000	LE2 1AB	Facial Palsy	876	2011-08-18 00:00:00.000	1899-12-30 08:52:00.000	Royal Marsden Hospital	Dr. Greg Knox	Muscle Relaxer
107	Female	1945-07-31 00:00:00.000	LE2 8YY	Cancer	9876	2012-12-20 00:00:00.000	1899-12-30 10:30:00.000	Royal Marsden Hospital	Dr. Coco Oji	Chemotherapy
100	Male	1945-07-31 00:00:00.000	LE1 6TB	Heart attack	6544	2020-05-02 00:00:00.000	1899-12-30 18:16:00.000	Freeman Hospital	Dr. House Beckingham	Bed Rest
250	Male	1945-07-31 00:00:00.000	LE3 4NN	Concussion	560	1991-05-13 00:00:00.000	1899-12-30 20:19:00.000	Deaconess Waltham Hospital	Dr. Nick Jean	Tylenol
250	Male	1945-07-31 00:00:00.000	LE3 4NN	Migraine	7865	1994-09-15 00:00:00.000	1899-12-30 13:32:00.000	Deaconess Waltham Hospital	Dr. Nick Jean	Paracetamol
250	Male	1945-07-31 00:00:00.000	LE3 4NN	Cough	456	1995-10-10 00:00:00.000	1899-12-30 15:30:00.000	Deaconess Waltham Hospital	Dr. Nick Jean	Coughlin
250	Male	1945-07-31 00:00:00.000	LE3 4NN	Chest pain	1234	1995-08-12 00:00:00.000	1899-12-30 21:15:00.000	Deaconess Waltham Hospital	Dr. Nick Jean	Nitroglycerin
250	Male	1945-07-31 00:00:00.000	LE3 4NN	Routine Chec...	3456	1993-09-21 00:00:00.000	1899-12-30 10:00:00.000	Deaconess Waltham Hospital	Dr. Nick Jean	Chloroquine
250	Male	1945-07-31 00:00:00.000	LE3 4NN	Routine Chec...	976	1996-05-31 00:00:00.000	1899-12-30 12:00:00.000	Deaconess Waltham Hospital	Dr. Nick Jean	No Prescription
250	Male	1945-07-31 00:00:00.000	LE3 4NN	Routine Chec...	235	1994-01-25 00:00:00.000	1899-12-30 17:30:00.000	Deaconess Waltham Hospital	Dr. Michael Dusty	Supplements
250	Male	1945-07-31 00:00:00.000	LE3 4NN	Routine Chec...	8753	1992-12-30 00:00:00.000	1899-12-30 12:45:00.000	Deaconess Waltham Hospital	Dr. Michael Dusty	Supplements
102	Female	1945-07-31 00:00:00.000	LE3 4NN	Headache	643	1996-05-18 00:00:00.000	1899-12-30 03:00:00.000	Bedford Trust Hospital	Dr. Hall Alley	Vegetables

Figure 4.6: Filtering *GIC* Dataset with the Governor's DoB

The result showed that only six individuals in the Governor's city shared the same date of birth, narrowing down the pool of potential entries that could be associated with the Governor. These six individuals with *Patient Codes* 100, 101, 102, 107, 109 and 250 include both the male and female gender. However, since it was also already known that the Governor is a man, the next query against the dataset was to filter the result to only return patients with the Governor's date of birth and those that are also male. This revealed that only three individuals in the GIC dataset have these two criteria in common, as shown in Figure 4.7.

```

1 SELECT *
2 FROM [dbo].[GICdata]
3 WHERE [Date of Birth] = '1945-07-31' and
4      [Gender] = 'Male'

```

	Patient Code	Gender	Date of Birth	Zip Code	Event	Event Code	Event Date	Event Time	Hospital	Attending Doctor	Prescription
1	101	Male	1945-07-31 00:00:00.000	LE2 7GP	Epileptic seizure	7645	2019-01-05 00:00:00.000	1899-12-30 05:32:00.000	Birmingham Children's Hospital	Dr. Who Laure	Paracetamol
2	100	Male	1945-07-31 00:00:00.000	LE1 6TB	Heart attack	754	2019-03-01 00:00:00.000	1899-12-30 14:59:00.000	Freeman Hospital	Dr. House Beckingham	Bed Rest
3	250	Male	1945-07-31 00:00:00.000	LE3 4NN	Influenza	3476	1996-05-18 00:00:00.000	1899-12-30 10:10:00.000	Deaconess Waltham Hospital	Dr. Nick Jean	Relenza
4	101	Male	1945-07-31 00:00:00.000	LE2 7GP	Epileptic seizure	564	2019-07-08 00:00:00.000	1899-12-30 05:00:00.000	Birmingham Children's Hospital	Dr. Who Laure	Paracetamol
5	100	Male	1945-07-31 00:00:00.000	LE1 6TB	Heart attack	563	2020-04-08 00:00:00.000	1899-12-30 01:32:00.000	Freeman Hospital	Dr. House Beckingham	Bed Rest
6	250	Male	1945-07-31 00:00:00.000	LE3 4NN	Diarhea	4556	1996-01-18 00:00:00.000	1899-12-30 12:02:00.000	Deaconess Waltham Hospital	Dr. Nick Jean	Loperamide
7	101	Male	1945-07-31 00:00:00.000	LE2 7GP	Epileptic seizure	876	2019-12-04 00:00:00.000	1899-12-30 10:15:00.000	Birmingham Children's Hospital	Dr. Who Laure	Paracetamol
8	100	Male	1945-07-31 00:00:00.000	LE1 6TB	Heart attack	6544	2020-05-02 00:00:00.000	1899-12-30 18:16:00.000	Freeman Hospital	Dr. House Beckingham	Bed Rest
9	250	Male	1945-07-31 00:00:00.000	LE3 4NN	Concussion	560	1991-05-13 00:00:00.000	1899-12-30 20:19:00.000	Deaconess Waltham Hospital	Dr. Nick Jean	Tylenol
10	250	Male	1945-07-31 00:00:00.000	LE3 4NN	Migraine	7865	1994-09-15 00:00:00.000	1899-12-30 13:32:00.000	Deaconess Waltham Hospital	Dr. Nick Jean	Paracetamol
11	250	Male	1945-07-31 00:00:00.000	LE3 4NN	Cough	456	1995-10-10 00:00:00.000	1899-12-30 15:30:00.000	Deaconess Waltham Hospital	Dr. Nick Jean	Coughlin
12	250	Male	1945-07-31 00:00:00.000	LE3 4NN	Chest pain	1234	1995-08-12 00:00:00.000	1899-12-30 21:15:00.000	Deaconess Waltham Hospital	Dr. Nick Jean	Nitroglycerin
13	250	Male	1945-07-31 00:00:00.000	LE3 4NN	Routine Check-up	3456	1993-09-21 00:00:00.000	1899-12-30 10:00:00.000	Deaconess Waltham Hospital	Dr. Nick Jean	Chloroquine
14	250	Male	1945-07-31 00:00:00.000	LE3 4NN	Routine Check-up	976	1996-05-31 00:00:00.000	1899-12-30 12:00:00.000	Deaconess Waltham Hospital	Dr. Nick Jean	No Prescription
15	250	Male	1945-07-31 00:00:00.000	LE3 4NN	Routine Check-up	235	1994-01-25 00:00:00.000	1899-12-30 17:30:00.000	Deaconess Waltham Hospital	Dr. Michael Dusty	Supplements
16	250	Male	1945-07-31 00:00:00.000	LE3 4NN	Routine Check-up	8753	1992-12-30 00:00:00.000	1899-12-30 12:45:00.000	Deaconess Waltham Hospital	Dr. Michael Dusty	Supplements

Figure 4.7: Filtering *GIC* Dataset with the Governor's DoB and Gender

AOL Data Re-identification

The focal point of this research is to analyse database search queries and the patterns in them that may lead to re-identification of a previously anonymised detail about a data subject. There is a limitation to the AOL data release scenario in regards to representing the re-identification story as SQL search queries. During analysis, this particular case looked extremely unrefined. This is due to the nature of this re-identification scenario. The process of with which re-identification occurred in this scenario does not naturally fit into the model that this research is exploring.

This scenario is about extracting the similarities in the search queries that point to a particular user (a user in the database that has complimentary searches). Searches with keywords are similar enough to assume that they are from the same user. This does not follow the technical aspects of re-identification that employs external database(s) to fill in the gaps in a database, gaps that are a result of deliberate anonymisation by the data controller.

4.4 Re-identification Attack Experiment Analysis and Results

4.4.1 Experiment Analysis

Netflix Prize Data

The interpretation of all the information gathered from querying the datasets involved combining two tables from the two databases and looking out for the similarities in the data that may hint that different data from different datasets are about the same individual. To attempt this, the resulting table shown in Figure 4.4 was joined with the one in Figure 4.5 for comparison. Since the movie column is already a common criterion, other columns (*Date of Grade*, *Grade*) from *TrainingData* and (*Ratings*, *Date*) from *IMDb* are used for comparison between the two databases.

From the *TrainingData* table, Figure 4.4 shows that *User 47* rated *Black Panther* on date *2004-04-22*, with a grade of *10*. This is consistent with the entry for user *tnoah@outlook.co.uk* in the *IMDb* table as shown in Figure 4.5. On *2004-04-08*, *User 7* gave a rating of *8*, and on the same day, a user with username *cliffordharris@outlook.co.uk* gave the same rating to the same movie. This is also the case for *User 43* from *TrainingData* and a user *Sprokopengo@gmail.com* in the *IMDb* table.

These three entries having the same *Date of Grade/Ratings* and being rated identically are the center of attention in the query outputs generated from the exploration of the dataset with SQL. It led to speculation that these entries from the two different datasets may be from the same individual. However, a single movie entry being identical is not a basis to conclude that these anonymous users from the Netflix Prize Data (*TrainingData*) had been re-identified in *IMDb*. To make this conclusion, a more elaborate exploration and analysis of the dataset had to be performed. For this, a *Re-identification Likelihood Quadrant* was generated, to realistically classify the probability of an entry about any particular movie being made by the same user on both platforms. The quadrant was created based on four columns in the two databases; (*Date of Grade, Grade*) from *TrainingData* and (*Ratings, Date*) from *IMDb*.

Q1 MOST LIKELY <i>Same Date</i> <i>Same Rating</i>	Q2 MORE LIKELY <i>Different Date</i> <i>Same Rating</i>
Q3 LESS LIKELY <i>Same Date</i> <i>Different Rating</i>	Q4 NOT LIKELY <i>Different Date</i> <i>Different Rating</i>

Figure 4.8: Re-identification Likelihood Quadrant
[76]

Q1, in Figure 4.8 represents entries that were made on the same date and awarded the same rating in both datasets. These are entries that have the highest likelihood of being made by the same individuals. User(s) that fall in this quadrant can be assumed, with a high probability that they have been re-identified. Q2 is for the class of users that awarded a movie the same rating, but on different dates on both datasets. There is a realistic presumption to be made about this being the same individual in both datasets, if the number of entries that satisfy this condition is quite significant for such a user. Q2 has a likelihood of being a re-identification scenario, depending on how prominent other determining factors are. The probability of the same person rating the same movie differently on different platforms is less, therefore the entries that satisfied this condition are in Q3, with very little likelihood of it being re-identification. The entries that fit the specifications of the last quadrant, present no sign of being by the same individual, they are assumed to be ratings entered by different users.

Governor William Weld

The re-identification in this scenario is a targeted re-identification. The strategy exploits all the details that were already known about the target of the attack. A series of SQL queries, as shown in Figures 4.6 and 4.7 was executed to filter the GIC dataset to reveal entries that may contain information about the target (Governor William Weld). Another available detail about this scenario is the date of the Governor’s health episode. Knowing this, a query to further filter the dataset using the *Event Date* criteria is executed. The result here showed a record of individuals who are males, shared the Governor’s date of birth, and had health event record entries on the same day as the Governor’s publicised hospitalisation.

After the data filtering of the GIC dataset based on already known information, the secondary *VoterRolls* dataset was then introduced, this is to reveal how close to the target the result is with the re-identification attack strategy being employed.

4.4.2 Experiment Results

Netflix Prize Data

With the quadrant in Figure 4.8 providing a template to classify re-identification likelihood based on the Date/Rating relationship shared by the users, queries targeting each of the three users highlighted in 4.4.1 were executed to examine how their ratings of other movies and the date they were done classifies them into any of the sections in the re-identification likelihood quadrant.

The results of the queries for *User 47*, *User 7* and *User 43* are shown in Figures 4.9, 4.10, and 4.11 below.

1

```
1 SELECT T.[User ID]
2 ,T.[Movies]
3 ,T.[Date of Grade]
4 ,T.[Grade]
5 ,I.[Usernames]
6 ,I.[Movies Titles]
7 ,I.[Ratings]
8 ,I.[Date]
9 FROM [dbo].[TrainingData] AS T join [dbo].[IMDb] AS I ON
10 T.[Grade]= I.[Ratings] and
11 T.[Movies]= I.[Movies Titles] and
12 T.[Date of Grade]= I.[Date]
13 WHERE T.[User ID]= 'user 7'
14 ORDER BY [User ID]}
```

¹T = TrainingData, I = IMDb

Results		Messages						
	User ID	Movies	Date of Grade	Grade	Username	Movie Name	Ratings	Date of Rating
1	user 7	Dexter	2004-05-03 00:00:00.000	6	cliffordharris@outlook.co.uk	Dexter	6	2004-05-03 00:00:00.000
2	user 7	Black Panther	2004-04-08 00:00:00.000	8	cliffordharris@outlook.co.uk	Black Panther	8	2004-04-08 00:00:00.000
3	user 7	Spider Man	2004-07-20 00:00:00.000	8	cliffordharris@outlook.co.uk	Spider Man	8	2004-07-20 00:00:00.000
4	user 7	Step Up	2007-08-02 00:00:00.000	7	cliffordharris@outlook.co.uk	Step Up	7	2007-08-02 00:00:00.000
5	user 7	Friends	2003-05-19 00:00:00.000	5	cliffordharris@outlook.co.uk	Friends	5	2003-05-19 00:00:00.000
6	user 7	The Big Bang Theory	2002-04-30 00:00:00.000	8	cliffordharris@outlook.co.uk	The Big Bang Theory	8	2002-04-30 00:00:00.000

Figure 4.9: Result for *User 7*

Results		Messages						
	User ID	Movies	Date of Grade	Grade	Username	Movie Name	Ratings	Date of Rating
1	user 47	Black Panther	2004-04-22 00:00:00.000	10	tnoah@outlook.co.uk	Black Panther	10	2004-04-22 00:00:00.000
2	user 47	Step Up	2005-08-09 00:00:00.000	5	tnoah@outlook.co.uk	Step Up	5	2005-08-09 00:00:00.000

Figure 4.10: Result for *User 47*

Results		Messages						
	User ID	Movies	Date of Grade	Grade	Username	Movie Name	Ratings	Date of Rating
1	user 43	Black Mirror	2004-03-29 00:00:00.000	8	Sprokopengo@gmail.com	Black Mirror	8	2004-03-29 00:00:00.000

Figure 4.11: Result for *User 43*

Figure 4.9 represents the comparison of movie rating entries on the two databases for “User 7”. The criterion of Q1 was satisfied for all of the entries made by this user. This is an indication of a high likelihood of re-identification. Figure 4.10 shows that the criterion of Q1 was satisfied for most, but not all entries, suggesting that there is a likelihood that this is a case of re-identification. “User 43” has minimal entry that fits into Q1, re-identification likelihood is low, as shown in Figure 4.11.

From this, it was concluded with a high likelihood that *User 7* from the Netflix *TrainingData* is the same individual as the user with the username *cliffordharris@outlook.co.uk*.

Governor William Weld (GIC Data)

As mentioned in 4.2.3, research shows that 87.1% of US citizens can be uniquely identified through their Date of birth, Zip Code, and Gender. These details were shared in the GIC dataset and can also be found in the Voter rolls. The *VoterRolls* table was then used as a secondary data source to attempt the re-identification of the Governor. The result from the SQL query used in the attempt is shown in figure 4.12 below.

```

1 SELECT *
2 FROM [dbo].[GICdata] as G
3 FULL OUTER JOIN [dbo].[VoterRolls] as V
4 on G.[Date of Birth] = V.[Birth Date] and G.[Zip Code] = V.[ZIP Code] and G.[
  Gender] = V.[Sex]
5 WHERE [Gender] = 'Male' and
6 [Date of Birth] = CONVERT(datetime, '1945-07-31') and
7 CAST([Event Date] AS datetime) = CONVERT(datetime, '1996-05-18')
```

Results	Messages	Patent Code	Gender	Date of Birth	Zip Code	Event	Event Code	Event Date	Event Time	Hospital	Attending Doctor	Prescription	Name	Address	ZIP Code	Birth Date	Sex
1		250	Male	1945-07-31 00:00:00.000	LE3 4NN	Influenza	3476	1996-05-18 00:00:00.000	1899-12-30 10:10:00.000	Deaconess Waltham Hospital	Dr. Nick Jean	Relenza	William Weld	10, singer street, Cambridge	LE3 4NN	1945-07-31 00:00:00.000	Male

Figure 4.12: Result for Re-identification Attempt

2

The result in Figure 4.12 showed that *Patient 250* in the GIC dataset is *William Weld* in the *VoterRolls*, which contained personal information about residents in the Governor’s town. This information filled in the gaps of the supposedly anonymised records of the shared GIC dataset.

4.5 Experiment Strategy

For the Netflix Prize data Scenario, the approach used while undertaking the re-identification in the experiment for this work was to isolate any user with properties that stood out (frequency of occurrence in this case) the most in both the *TrainingData* and the IMDB dataset. After establishing distinctive properties from the datasets, the next step was cross-referencing these distinctive properties between the two datasets and analyse the result for re-identification clues. Recognising these properties and relating them to a secondary dataset are the two main stages involved in this strategy. The objective of the re-identification attack demonstrated in this scenario is *existential*, as it aims directly towards proving that some user from the anonymised Netflix *TrainingData* is re-identifiable. There was no prior knowledge about any individual in the dataset, making it impractical to target any specific user. However, a targeted approach aiming to re-identify any specific record rather than a specific user could also be feasible.

Strategically, another re-identification objective that could also be applicable in this dataset scenario will be *universal* re-identification. This is because the re-identification in the dataset could be on a larger scale. However, to attempt this, a change in the attack strategy will be imperative. The strategy for universal re-identification will focus less on making out distinctive users. Instead, attack efforts will focus on attempting to re-identify as many as possible individuals from the dataset.

The Governor William Weld scenario shared some similar approaches as the Netflix scenario, it aimed at establishing some unique properties first, then cross-referencing with a secondary dataset. However, the attack strategy in this scenario was armed with some prior information about the target. The objective of the re-identification attack demonstrated is *targeted*, aiming towards a single individual. Prior information available about the Governor had already placed him in the GIC dataset and the availability redundant information relating to the GIC dataset made the attack possible.

²G = GICdata, V = VoterRolls

4.6 SQL Query Data Collection

After the results from experimenting with re-identification on different datasets, this research concluded that re-identification can be represented as a series of database queries. The exploration study and re-identification attack re-creation in this chapter detail a series of SQL database queries used to technically achieve re-identification. This notion became the basis on which the approach for research data collection was developed. The aim of the data collected for this research was to gather user queries against an anonymised database.

In order to design an appropriate experiment and achieve reliable results for the aim of this research work, a data collection campaign had to be launched. This project aims to ascertain that re-identification attacks are recognisable by studying the patterns in SQL queries being executed on anonymised databases. Therefore, a collection of real user SQL database queries is essential to achieve the project aim. Using the synthetic dataset created in 4.3.2 for the Netflix Prize Data scenario, a data collection event was organised. The event involved users with beginner, intermediate and advanced skills in the use of SQL for database queries.

4.6.1 Data Collection Campaign

A Windows 10 virtual machine (VM) was created for this campaign. Microsoft SQL Server Management Studio (SSMS) software was installed on the VM and the synthetic database for Netflix Prize Data was uploaded onto the SSMS. Then the Virtual Machine was exported to a *.ova* (Open Virtual Appliance) file to be shared among the participants. Two versions of the database were uploaded; labelled *ALPHA* and *BETA*. The *ALPHA* version consisted of only the Netflix Prize Data tables while the *BETA* version included the IMDb database as a secondary data source. The *ALPHA* database is to be engaged by participants with the aim of executing and generating regular SQL queries users will normally employ on a day-by-day basis. Also, another purpose of this step in the activity is to allow participants the opportunity to get familiarised with the structure of the database. The *BETA* database sets the grounds for different re-identification attack attempts to be made by participants, as this database includes an external open-source database (IMDb) among its tables.

The data collection activity was broken down into two sessions, with each of the sessions aimed at achieving different goals.

- **Session One:** This session provided general background information about the research, covering concepts such as anonymisation and re-identification with the participants. The re-identification story of Governor William Weld was used to get participants thinking about how a re-identification attack may be approached conceptually and practically, through the reverse engineering of Latanya Sweeney's thought process and actions in her execution of the attack against the Governor.

Also during the session, the ethical implication of the activity about to be engaged in was discussed. The information shared with participants in regard to the ethics of the campaign is as follows:

1. According to the Data Protection Act 2018, performing re-identification is illegal, except for research purposes.

2. The research is requesting participants' permission to their SQL query data, and the recorded queries are meant to be used to support the research work.
3. The data collection will record SQL query data only. No personal information is recorded. The recorded query data will not be relatable to any participants.
4. Participants have the option to choose whether their SQL query data is included for use in further research analysis.

The complete details of the information (experiment guide) shared with the participants in the data collection activity can be found in A.2.

This session was closed by referring participants to an instructional YouTube video [75]. The video was created specifically for this data collection activity, and it demonstrates the use of the software environment to be used for database access, query execution, and query recording.

- **Session Two:** The second session involved the participants getting familiar with the database in use for the data collection activity (*Netflix Prize Data ALPHA*). Using SQL queries, participants engaged with the database in an attempt to learn as much as possible about the database's structural details. Familiarity with the database structure was presumed by the research to be able to help participants come up with creative SQL queries and approaches to attempt re-identification later on in the activity.

SQL query data collected during this session will serve as the benchmark for what "normal" (non re-identification attempt) database queries are like. This will be useful in the eventual training of a Machine Learning Model.

- **Session Three:** For the final session, the specifics of the research and the expectation from the session was introduced. Participants were required to combine their technical SQL skills with the information about re-identification provided in the first session to attempt a re-identification attack against the database. The *Netflix Prize Data BETA* database, with the IMDb secondary database included was used at this junction, to attempt linkage of the individual(s) that may be the same person(s) from both databases. Participants were encouraged to execute as many as possible SQL query ideas and approaches participants think may be appropriate to achieve the aim of this session.

4.6.2 Data Collection Tools and Setup

On the Virtual Machine created for the data collection activity, the Microsoft SSMS tool was installed and connected to a local SQL server. SQL Server Management Studio (SSMS) is an integrated platform for maintaining a SQL infrastructure (SQL Server) [103]. SSMS is equipped with tools to configure and monitor instances of SQL Server and databases. With the SSMS environment, this research and the participants in the data collection activity are able to write and execute SQL queries against a database. An essential part of this data collection activity is the recording of the queries that participants are executing. The Microsoft SSMS suite has a built-in tool that serves this purpose. This tool is known as *Microsoft SQL Server Profiler*.

Microsoft SQL Server Profiler is a Graphical User Interface (GUI) tool for monitoring, capturing, and saving data about each SQL event to a file for later inspection. It can be used to record and analyse stored procedures (a set of SQL queries stored to be reused on a database)

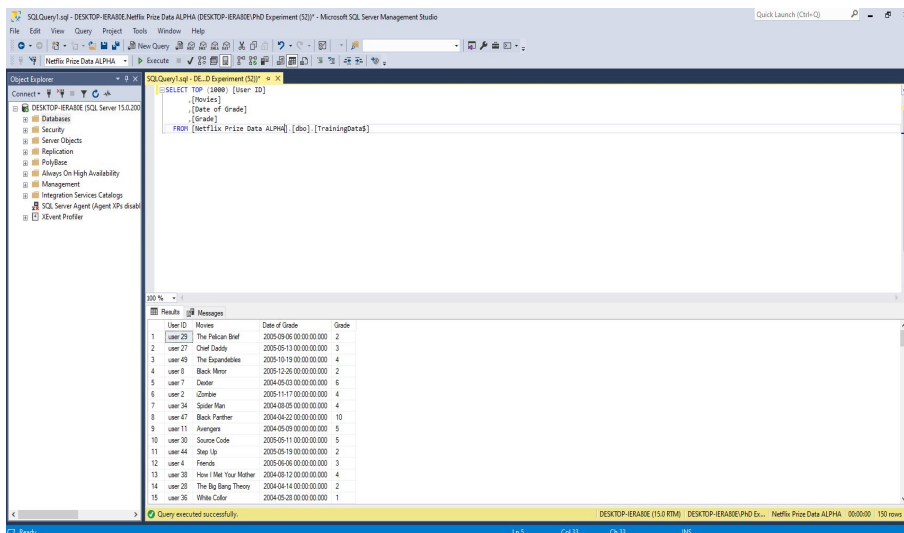


Figure 4.13: Microsoft SSMS

that may be adversely affecting the performance in a SQL environment [104]. This research work employed the functionalities of SQL Server Profiler to trace and capture the SQL queries that were being executed by the participants in the data collection event. The recorded queries are the data being collected and the focal point of the event.

The screenshot shows the Microsoft SQL Server Profiler interface with a detailed trace of SQL events. The table below represents the data shown in the trace:

EventClass	TextData	ApplicationName	NTUserName	LogName	CPU	Reads	Writes	Duration	ClientProcessID	SPID	StartTime	EndTime	BinaryData	DatabaseID	DatabaseName
Trace Start											2021-01-27 20:51:26...				
ExistingConnection										51					1 master
ExistingConnection										52					1 master
ExistingConnection										53					6 NetfliX P...
ExistingConnection										54					1 master
SQLBatchStarting	SET LOCK_TIMEOUT 10000	Microsoft SQ...	PIH Exp...	DESKTO...						3324	56	2021-01-27 20:51:36...			1 master
SQLBatchCompleted	SET LOCK_TIMEOUT 10000	Microsoft SQ...	PIH Exp...	DESKTO...	0	0	0	0		3324	56	2021-01-27 20:51:36...			1 master
SQLBatchStarting	DECLARE @edition sysname; SET @edit...	Microsoft SQ...	PIH Exp...	DESKTO...						3324	56	2021-01-27 20:51:36...			1 master
SQLBatchCompleted	DECLARE @edition sysname; SET @edit...	Microsoft SQ...	PIH Exp...	DESKTO...	0	92	0	10		3324	56	2021-01-27 20:51:36...			1 master
RPCCompleted	exec sp_executesql N'SELECT CAST(HAS...	Microsoft SQ...	PIH Exp...	DESKTO...	15	1145	0	58		3324	56	2021-01-27 20:51:36...	0X000000...		1 master
RPCCompleted	exec sp_executesql N'SELECT ISNULL(H...	Microsoft SQ...	PIH Exp...	DESKTO...	0	0	0	0		3324	56	2021-01-27 20:51:36...	0X000000...		5 NetfliX P...
SQLBatchStarting	IF OBJECT_ID (N'[sys].[database_quer...	Microsoft SQ...	PIH Exp...	DESKTO...						3324	56	2021-01-27 20:51:36...			5 NetfliX P...
SQLBatchCompleted	IF OBJECT_ID (N'[sys].[database_quer...	Microsoft SQ...	PIH Exp...	DESKTO...	16	2675	0	35		3324	56	2021-01-27 20:51:36...			5 NetfliX P...
RPCCompleted	exec sp_executesql N'SELECT dtb.col1...	Microsoft SQ...	PIH Exp...	DESKTO...	31	62	0	29		3324	56	2021-01-27 20:51:43...	0X000000...		1 master
SQLBatchStarting	SELECT dtb.name AS [Name], dtb.datab...	Microsoft SQ...	PIH Exp...	DESKTO...						3324	56	2021-01-27 20:51:43...			1 master
SQLBatchCompleted	SELECT dtb.name AS [Name], dtb.datab...	Microsoft SQ...	PIH Exp...	DESKTO...	15	92	0	19		3324	56	2021-01-27 20:51:43...			1 master
SQLBatchStarting	SET LOCK_TIMEOUT 10000	Microsoft SQ...	PIH Exp...	DESKTO...						3324	56	2021-01-27 20:51:45...			1 master
SQLBatchCompleted	SET LOCK_TIMEOUT 10000	Microsoft SQ...	PIH Exp...	DESKTO...	0	0	0	0		3324	56	2021-01-27 20:51:45...			1 master
SQLBatchStarting	DECLARE @edition sysname; SET @edit...	Microsoft SQ...	PIH Exp...	DESKTO...						3324	56	2021-01-27 20:51:45...			1 master
SQLBatchCompleted	DECLARE @edition sysname; SET @edit...	Microsoft SQ...	PIH Exp...	DESKTO...	0	0	0	1		3324	56	2021-01-27 20:51:45...			1 master
SQLBatchStarting	use [NetfliX Prize Data BETA]	Microsoft SQ...	PIH Exp...	DESKTO...						3324	56	2021-01-27 20:51:45...			1 master
SQLBatchCompleted	use [NetfliX Prize Data BETA]	Microsoft SQ...	PIH Exp...	DESKTO...	0	0	0	0		3324	56	2021-01-27 20:51:45...			5 NetfliX P...
RPCCompleted	exec sp_executesql N'SELECT tdl.name...	Microsoft SQ...	PIH Exp...	DESKTO...	157	3584	10	198		3324	56	2021-01-27 20:51:45...	0X000000...		5 NetfliX P...
SQLBatchStarting	SET LOCK_TIMEOUT 10000	Microsoft SQ...	PIH Exp...	DESKTO...						3324	56	2021-01-27 20:51:45...			1 master
SQLBatchCompleted	SET LOCK_TIMEOUT 10000	Microsoft SQ...	PIH Exp...	DESKTO...	0	0	0	0		3324	56	2021-01-27 20:51:45...			1 master
SQLBatchStarting	DECLARE @edition sysname; SET @edit...	Microsoft SQ...	PIH Exp...	DESKTO...						3324	56	2021-01-27 20:51:45...			1 master
SQLBatchCompleted	DECLARE @edition sysname; SET @edit...	Microsoft SQ...	PIH Exp...	DESKTO...	0	0	0	2		3324	56	2021-01-27 20:51:45...			1 master
SQLBatchStarting	use [NetfliX Prize Data BETA]	Microsoft SQ...	PIH Exp...	DESKTO...						3324	56	2021-01-27 20:51:45...			1 master
SQLBatchCompleted	use [NetfliX Prize Data BETA]	Microsoft SQ...	PIH Exp...	DESKTO...	0	0	0	1		3324	56	2021-01-27 20:51:45...			5 NetfliX P...
RPCCompleted	exec sp_executesql N'SELECT tdl.name...	Microsoft SQ...	PIH Exp...	DESKTO...	140	417	0	157		3324	56	2021-01-27 20:51:45...	0X000000...		5 NetfliX P...
SQLBatchStarting	SET LOCK_TIMEOUT 10000	Microsoft SQ...	PIH Exp...	DESKTO...						3324	56	2021-01-27 20:51:45...			1 master
SQLBatchCompleted	SET LOCK_TIMEOUT 10000	Microsoft SQ...	PIH Exp...	DESKTO...	0	0	0	0		3324	56	2021-01-27 20:51:45...			1 master

Figure 4.14: Microsoft SQL Server Profiler

SQL Server Profiler Trace Configurations

Configuration settings in the SQL Server Profiler tool were adjusted to fit the types of information that the research expects SQL queries should have, capturing the details about SQL queries being executed by participants on the database. The captured information will then be present in query data for all the instances of tracing that get recorded for the data collection. This creates a level of

consistency in the features captured about the queries, regardless of how different the approaches in query executions are from one participant to another.

To make these configuration adjustments, the parameters of the *Trace Properties* for every instance of the capture had to be specific. The *Trace Properties* in SQL Server Profiler has two tabs to configure, the *General* and the *Events Selection* tabs.

1. **General:** The *General* tab for the *Trace Properties* in SQL Server Profiler is where the basic settings are configured. These settings include the file name, file size to accommodate the query data to be captured, and other standard SQL Server settings, as shown in Figure 4.15.

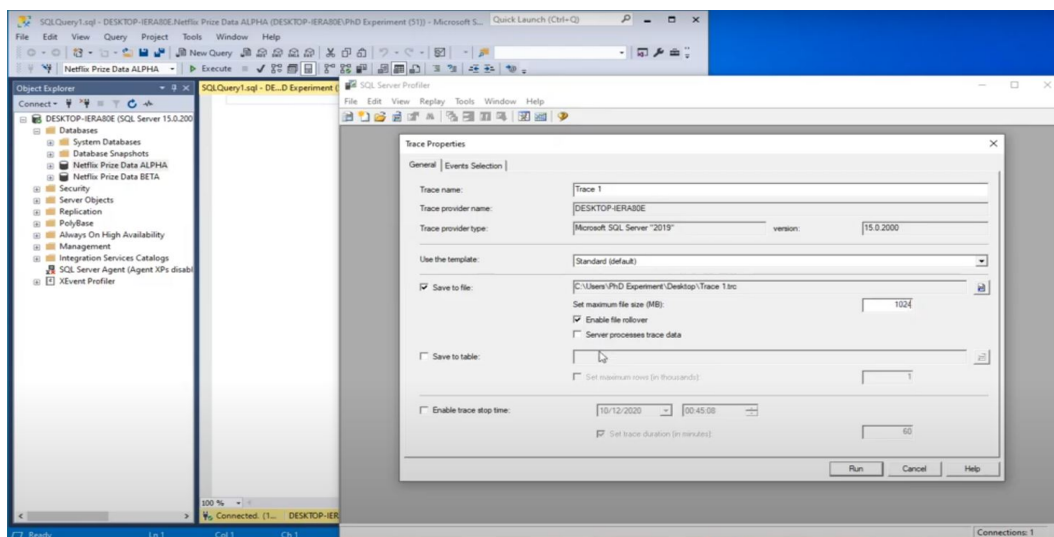


Figure 4.15: *General* Tab Configurations for the Trace Properties

2. **Events Selection:** This tab is where discretionary decisions were made about what details the trace function records about participants' queries during the data collection activity. The capture of events relating to the Security Auditing function of the SQL Server Profiler was disabled for the traces, as these are irrelevant to the research goal. Trace settings that were enabled were those that relate to capturing queries executed by the participants. The video resource at [75], created for this data collection activity shows the complete details of the parameters set for usage in SQL Profiler configuration. The details enabled to be captured by the trace were streamlined to focus on those that relate to executed queries. This is to reduce the amount of noise (irrelevant information) captured in the trace.

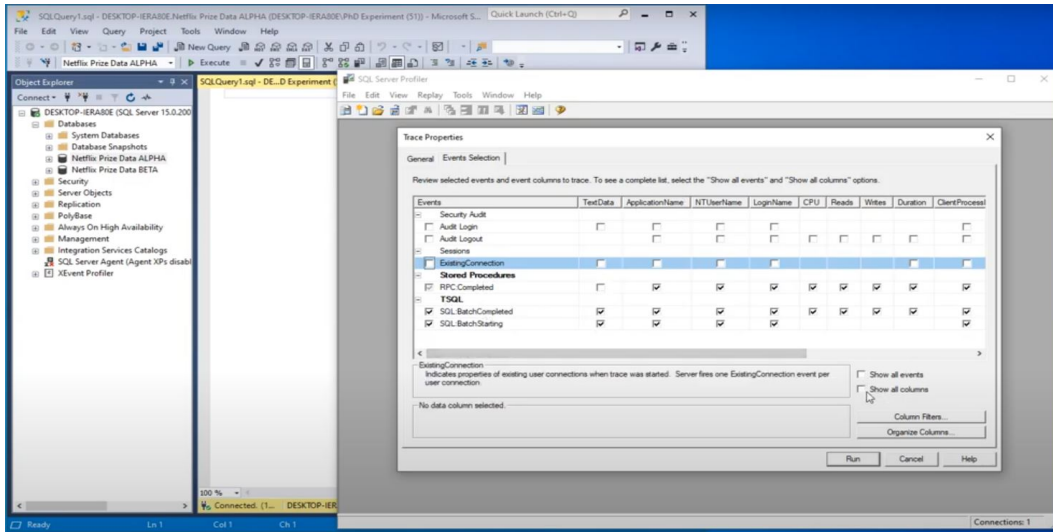


Figure 4.16: *Events Selection* Tab Configurations for the Trace Properties

With these settings configured, a trace was then initiated for every query attempt the participants engaged in. This was done for both the re-identification attempt queries and the normal queries. The trace runs in the background while queries were being executed in the SSMS and the trace is stopped once the participant is done with running various query ideas. The trace is automatically saved to the file created on the *General* tab of the *Trace Properties* (the file was created before the trace started).

4.7 Data Augmentation

Machine Learning algorithms rely on big data to a great extent. The more data points an algorithm can learn from in the training dataset during the training phase of developing a machine learning model, the better the performance of the model. A large amount of data help machine learning models avoid *overfitting* problem. The machine learning models in this work are trained towards recognising patterns in unseen data points and achieving high performance while doing so. However, a model is trained on a set of training data on which its performance will not be evaluated. As mentioned in 3.7, overfitting is a case whereby the model memorises the different characteristics of training data verbatim, as opposed to studying and learning the discipline behind the data, to formulate a general predictive rule. Overfitting occurs when there is noise in the dataset and a shortage of datasets to train the model [142, 36, 168]. Training data deficiency is one of the main challenges in machine learning tasks [56], and data augmentation methods provide a solution to the shortage of dataset challenges when training a machine learning algorithm. Data augmentation consists of a suite of techniques that supplement the quantity and quality of the training dataset. This allows the development of an optimally trained machine learning model [142].

The data collection for this research presented in 4.6 requires a practical skill (knowledge of SQL programming language) from the participants. This reduces the research's ability to gather SQL query data in large amounts. There was very limited data available to train machine learning algorithms with, at the end of the data collection event. To avoid the pitfalls of building a model

with minimal datasets, such as overfitting, this research adopted and implemented principles of data augmentation to enhance the amount of datasets for the training of algorithms that can recognise re-identification patterns in database queries. Data augmentation is implemented for this research dataset using Python as presented in 5.2.3.

As detailed in section 5.2.2, the data mining experiment in this research implemented the Bag-of-words (BoW) model at the data preprocessing stage. As such, the data augmentation methods adopted in this work are those that are applicable at the *word level*. Text data augmentation can also be applied at the character, phrase, sentence, and document level [11]. Due to the frustration from the inadequate performance of text classification models trained on a limited dataset, Wei and Zou in [162] proposed a set of techniques referred to as *Easy Data Augmentation* (EDA). EDA involves word replacement-based data augmentation techniques, it consists of four simple but powerful operations that have been proven to improve performance on text classification problems involving an insufficient amount of dataset [162, 92]. The data augmentation methods implemented in the machine learning experiment for this research are based on EDA, which used four operations including *Synonym Replacement* (SR), *Random Insertion* (RI), *Random Swap* (RS), and *Random Deletion* (RD). The data augmentation in this work implements Synonym Replacement and Random Insertion.

1. **Synonym Replacement (SR):** Generally, this involves randomly choosing n words from a dataset and replacing each of these words with one of its synonyms selected at random. In standard SR operation, a word (token) is gets replaced randomly by one of its synonyms retrieved from WordNet (a large word database of English words including nouns, adjectives, adverbs, and verbs). The standard SR operation is used in Natural Language Processing (NLP), however, the dataset in this research is SQL queries (computer language). Therefore, instead of replacing a word with its synonym, the word gets replaced by a *MASK* token. This is referred to as *random word masking* [124, 34]. The implementation of random word masking in this research is presented in 5.2.3.
2. **Random Insertion (RI):** In this EDA technique, the random insertion data augmentation operation involves randomly selecting a token (word) from the input data and inserting one of its synonyms from the WordNet word database to a random position in the input dataset. Similar to the approach used for random word masking, the RI augmentation experiment in this research randomly inserts tokens into the dataset. This approach is referred to as *random token insertion* [124].
3. **Random Swap (RS):** This method involves randomly choosing two tokens in the dataset and swapping their positions. This is then repeated n times.
4. **Random Deletion (RD):** This EDA technique involves randomly removing each word in the dataset with probability p .

This research implements Synonym Replacement and Random Insertion because their operation fits the machine learning experiment approach for the research dataset, as the research dataset is labeled and to be explored as a classification problem. Cases of random swap and random deletion methods have been reported in the literature to not preserve the dataset labels after augmentation [172, 5, 175, 97].

4.8 Chapter Summary

Chapter 4 presents the methodology that this research work used in the achievement of its main milestones, which is to ascertain that re-identification attacks can be a product of a series of SQL queries. Analysis of the different real-life data re-identification scenarios used as case studies is detailed in this chapter. The 2006 AOL data release, the re-identification of Netflix Prize Data, and Governor William Weld were explored for the purpose of this research. The exploration involved generating synthetic datasets for different scenarios and experimenting with the dataset to establish if re-identification can be recognised as a series of SQL queries. This chapter demonstrates the techniques, strategies, and queries used to achieve re-identification of the synthetic data. This was done for the Netflix Prize Data and Governor William Weld (GIC Data) scenarios. Studying the AOL scenario, this research concluded that the re-identification case there did not fit into the model that this research is exploring. The AOL scenario did not involve the use of a secondary database to exploit weak anonymisation techniques and the misguided release of a de-identified database.

The results of the experiments presented in this chapter helped this research arrive at a conclusion that re-identification attacks on anonymised databases are achievable through a series of database queries. Analysis of the results is presented based on the level of likelihood that any particular individual has been re-identified. A *re-identification Likelihood Quadrant* is created by the research for the Netflix Prize Data scenario, to measure the probability of the success of re-identification attempts. The experiment concludes that three individuals from the anonymised synthetic Netflix Prize Dataset; *User 47*, *User 7*, and *user 43*, are re-identified. However, with varying levels of likelihood. The experimentation and result of the synthetic GIC dataset also led this research to conclude that an individual in the GIC synthetic dataset was re-identified, based on the match of the Date of birth, Zip Code, and Gender from the anonymised GIC dataset and a secondary database.

The experiment emphasised the exploitation of redundant records that exist in the anonymous dataset and secondary (publicly available) datasets. This chapter analysed and discussed the re-identification experiments in relation to the objectives. With the Netflix Prize Data, the re-identification objective is *existential*. The aim is directed towards proving that there is a re-identifiable individual in the dataset. Even though the Governor William Weld GIC data scenario shares some strategic similarities with the Netflix case, the objective of re-identification in this instance is *targeted*. Governor William Weld is singled out to be re-identified.

With the results and conclusion from experimenting with re-identification attacks, the next phase of this research is also presented in this chapter. This involved organising a campaign for the collection of SQL database queries. Participants in the data collection campaign engage with the synthetic datasets by executing non re-identification attempt (*normal*) queries to get familiar with the database structure and then subsequently execute queries to attempt re-identification attacks. These SQL queries were recorded to be used in a data mining experiment, as part of the process of achieving the aim of this research work.

Chapter 5

Data Mining Models for Re-identification Query Pattern Recognition

The previous chapter presented a re-identification attack exploratory study that analysed different re-identification attack scenarios. Technical re-creation of the attacks and dataset collection were also described. In this chapter, the data mining experiment and application of machine learning algorithms for re-identification query pattern recognition are presented. The approach applied in this research to process the dataset for the training of machine learning algorithms is described in section 5.2. The training process for the four machine learning algorithms, including the strategy employed for training is detailed in sections 5.3 and 5.4. This chapter covers the Python implementation for the data mining experiment conducted in this research, including the algorithms used for data preparation detailed in section 5.2.2, data augmentation in section 5.2.3, and feature extraction in section 5.2.4.

5.1 Data Mining Experiment Overview

At the end of the data collection campaign embarked on in section 4.6, all the data collected as a result of the engagement with the database by participants in the data collection campaign were saved. Every instance of a set of normal (non re-identification) and re-identification attack attempt queries executed by the participant are traced and saved into a *trace file (.trc)* by the SQL server profiler. The next step towards achieving the aim of this research is to begin an analysis of the collected SQL query data. As mentioned in 3.3.2, raw datasets of various types and formats can be analysed and processed to infer meaningful connections through various algorithms that are suitably implemented in Python libraries. The aim of the work in this chapter is to process the research dataset and convert it to a format suitable for training machine learning models, and then train different binary classification algorithms to automatically recognise re-identification patterns in a SQL query being executed on a database. This involved experimenting with the SQL query dataset and exploring data mining techniques as well as machine learning algorithms in the Python library that are applicable to the problem domain of this research.

As a result of the data collection campaign, fourteen sets of SQL query data were gathered in the form of SQL trace files. Seven of the files represent the non re-identification query entries

and the other seven contain the re-identification attack attempt query entries. The *.trc* trace files were converted to *.sql* query files. The contents of the SQL files for one instance of both a normal query and a re-identification attack attempt query are as shown in appendices B.1 and B.2 respectively. The SQL files were further converted to a Comma-Separated Values (CSV) file format (*.csv*), to allow easy use in the Python implementation. Python has built-in libraries that allow manipulation and analysis of datasets in a *.csv* file format.

To train and develop an effective classifier model for the recognition of re-identification attempt query patterns, a significant amount of data samples is required in the training of such a model. The more the quantity of available data samples the model can learn from, the higher the chances of the classifier making accurate predictions. The available fourteen query samples generated from the data-collecting campaign are not sufficient to train and develop an accurate model. Therefore, this research implemented *data augmentation* techniques to generate more SQL query data samples. The data augmentation process was achieved using Python and it is detailed in 5.2.3. This chapter presents the Python implementation in training and building a classifier model for data re-identification attack attempt query recognition.

5.2 Data Preprocessing

SQL is a query language used to interact with databases and each CSV file containing SQL query data generated in the data collecting stage of this work holds a vast amount of SQL query information. The amount of information in each file is a reflection of user and system activities recorded by the SQL server profiler, these activities include SQL queries executed and any system information recorded in the background automatically by the profiler. This research handled the SQL dataset with an assumption that every data point has the possibility of contributing to patterns that makes a re-identification attempt query or a normal query. Details on how the dataset was prepared for analysis are presented in section 5.2.2. Data preprocessing is an essential stage of a data mining experiment and must be done prior to any of the other stages. This stage involved exploring different text preprocessing techniques and then transforming the preprocessed data in a format that makes it suitable to input into other algorithms. The Python programming language is equipped with libraries that are applicable in data processing and this research employed the functionalities of these Python libraries to approach its data preprocessing.

5.2.1 SQL Query Data Preprocessing Libraries

Python libraries are imported to allow the necessary functionalities required to manipulate the SQL dataset and transform it to a more suitable state and format (a format compatible with the machine learning algorithms to process). Six Python libraries are imported to be used for the preprocessing activity required to prepare the SQL dataset. These libraries are *re* (Regular Expression), *os* (Operating System), *pandas*, *pickle*, *random* and *numpy*. The regular expression library is used to find and match patterns in the texts of the SQL dataset. Importing the Operating System library is required to allow interaction with the system's local files and directories. This allows the *.csv* files containing the SQL data to be uploaded and used within the Python implementation. Pandas is a Python library that provides data preprocessing, manipulation, and analysis functions. Pandas data frames have three components: rows, columns, and data. The pickle library allows the conversion of a dataset object into a byte stream and saved it to a file to be used at a later time. Random is a library used for randomisation of contents of a dataset,

used for data augmentation in section 5.2.3 and Numpy is a library in Python used for working with Python arrays and lists.

5.2.2 Data Preparation and Labelling

Text data (i.e. SQL query dataset) has an unstructured data format, making it difficult for machine learning algorithms to process. Therefore, the text dataset must be converted to numbers. An effective method for initiating this conversion is the *Bag-of-Words (BoW)* model (an array data structure). The BoW model ignores the order of the information in the dataset, but treats features in the textual dataset as *words*. The representation of dataset information as words is referred to as *tokenization*. The BoW model operates using the frequency of occurrence of each word (token) representation in the dataset [20]. The data preparation stage involved transforming the dataset into a format that will improve the quality and relevance of the patterns that machine learning algorithms learn from the dataset. If a raw dataset that includes a significant amount of irrelevant data (noise) is used in the research data analysis, the result of the analysis will either be skewed or suboptimal. This is also known as “garbage in, garbage out”. A poorly prepared dataset will teach machine learning algorithms bad patterns. The data mining experiment in this research is a supervised learning task, as established in section 3.6, therefore, datasets are assigned labels to signify class membership. Dataset labelling is implemented as enumerated in step 3 of Algorithm 2. The Python implementation for this algorithm is in Appendix C.1.

Algorithm 2 Data Preparation and Labelling Algorithm

1. Create a regular expression pattern using Python raw string notation
 2. Specify a function to read and label the dataset from *.csv* files
 3. Set class label to *0* for normal query files and *1* for re-identification attempt query files
 4. Iterate through the rows, columns, and cells of *.csv* files. Such that if cell data is valid (not null), it is converted to a string and then stored to a *list_of_words* variable
 5. Use regular expression pattern to add space around special characters
 6. Split spaced-out row data into words and add to *list_of_words*
 7. Iterate through *list_of_words* and add words to a new *clean_words* variable. Ignore null spaces, removing “ ’ ” and replacing numbers with $\langle NUM \rangle$
 8. Return *clean_words* and labels
 9. Append a variable *X* to *words* and *Y* to *labels*
 10. Generate a tuple of *X* word list, *Y* label list and store in a new variable *data*
 11. Save *data* to file as *preprocessed_dataset* in pickle file format
-

5.2.3 SQL Query Data Augmentation

As expressed in section 4.7, the limited amount of SQL query data gathered for this research makes it imperative to perform data augmentation. The more data points the classifier models can study and learn from, the better their ability to recognise re-identification patterns in database queries. Data augmentation of the research dataset is achieved with the use of data generation algorithms implemented in Python. Data slicing (selecting part of a dataset, therefore breaking it down into smaller pieces) and randomisation techniques in Python are used in the augmentation process. Section 4.7 presented synonym replacement (random word masking) and random insertion (random token insertion) as the data augmentation techniques to explore in this research experiment. Appendices C.1.2 and C.1.4 depict the word cloud representation of the research dataset after random word masking and random token insertion respectively.

Random Word Masking

The overall strategy adopted for the random word masking augmentation is replacing a random 5% of words in the dataset documents with a “<MASK>” token. This is repeated 100 times, to generate a 100 new data structure representation for each 1,000-length data sequence with the same label. The dataset contains 14 *.csv* files (7 per class) with a total of 117,177 words and 1,928 unique words. At the end of the random word masking data augmentation process presented in Algorithm 3, the dataset is augmented to contain 11,110 files (5,555 per class). The Python implementation of this algorithm is presented in Appendix C.1.1.

Algorithm 3 Random Word Masking Augmentation Algorithm

1. Load the *preprocessed_dataset* for reading in binary format and save it to a *data* variable
 2. Set the sequence length for data slicing to a *fixed_length* of 1,000
 3. Declare new variables to store the *augmented_data* and the *label*
 4. Iterate through each document in *data* to get the *words* and the *label*
 5. Declare a variable to store the full list of words in the documents
 6. Declare a *current_range*, then a while loop that uses the *current_range* to check if there is a word length lower than the *fixed_length* of 1,000 in the full word list. If there are, the loop breaks
 7. Get 1,000 *sliced_words* and add to the *current_range*
 8. Append *sliced_words* list to the *augmented_data* variable created in step 3 and append the *label*
 9. Randomise by creating 100 new data points with 1,000 *fixed_length* and randomly mask 5% of words to create different structure in the dataset
 10. Iterate through the random list and replace the random 5% of words with the token *<MASK>*
 11. Append the new data structure to initially declared *augmented_data* and add the *label* variables
 12. Update *current_range* to point to the next 1,000-word range
-

Random Token Insertion

For the random token insertion augmentation approach, 5% tokens (words) are to be randomly inserted in the dataset. The data augmentation takes two functions, which are the original data and the insertion percentage. The function loops through each of the 14 *.csv* files, and for each of the files, the function slices the data in the file into fixed-length (1000-length) segments. 100 new data structure is generated for each segment by randomly inserting tokens. The function then returns the augmented data and labels. After the implementation of random token insertion as presented in algorithm 4, the augmented dataset contains 7,373 files and labels. The Python implementation of this algorithm is presented in Appendix C.1.3.

Algorithm 4 Random Word Insertion Augmentation Algorithm

1. Load the *preprocessed_dataset* for reading in binary format and save it to a *data* variable
 2. For each file, slice the data into *fixed_length* segments of 1,000
 3. Declare new variables to store the *augmented_data* and the *label*
 4. Iterate through each document in *data* to get the *words* and the *label*
 5. Declare a variable to store the full list of words in the documents
 6. Declare a *current_range*, then a while loop that uses the *current_range* to check if there is a word length lower than the *fixed_length* of 1,000 in the full word list. If there are, the loop breaks
 7. Get 1,000 *sliced_words* and add to the *current_range*
 8. Append *sliced_words* list to the *augmented_data* variable created in step 3 and append the *label*
 9. Randomise by creating 100 new data points with 1,000 *fixed_length* and randomly insert 5% of tokens to create different structure in the dataset
 10. Iterate through the random list and inserts the random 5% of tokens from WordNet into the dataset
 11. Append the new data structure to initially declared *augmented_data* and add the *label* variables
 12. Update *current_range* to point to the next 1,000-word range
-

5.2.4 Feature Extraction

Feature extraction is the process of representing or encoding words in a format that a machine learning algorithm is able to process, the process transforms the words in the SQL dataset files into numerical features. As shown in Algorithm 2, the BoW representation of the SQL dataset creates a vocabulary of words from the dataset and a measure of the frequency of occurrence of these words. This method is implemented to transform the textual tokens from the dataset into *features* (a unique representation of all known words in the dataset). The dataset representation generated by the feature extraction process is then used in training a machine learning algorithm. There are different techniques available for feature extraction in the Python machine learning library. Two of the most prominent algorithms implemented in textual data processing are implemented in this work. These are the *Tokenizer* algorithm and *TF-IDF* (Term Frequency - Inverse Document Frequency) algorithm.

Similar to the BoW methods, which represent the dataset as a collection of words (ignoring the order and structure) and as a vector of word frequencies, TF-IDF also considers the frequency of a token (term frequency) in the dataset. However, TF-IDF also considers the importance of the term in the entire dataset (inverse document frequency). A more significant weight is assigned to

terms that appear frequently in a document (file) but appeared less in the entire dataset. With this process, TF-IDF captures the discriminative power of terms (tokens/words) by attributing more weight to rare and important terms. TF-IDF is recognised to be advantageous over methods that are solely frequency-based such as Bag-of-Words [137].

To implement *Tokenizer* feature extraction in Python, functionalities of the *keras* Python framework are employed. The data preprocessing function in Python known as *Tokenizer* is used for the representation of the tokens in the research dataset, as shown in Algorithm 5. For TF-IDF, the *TfidfVectorizer* is imported from the *sklearn* library in Python.

***Tokenizer* Feature Extraction**

The *Tokenizer* function is used to create a vocabulary index based on the frequency of words in the list of dataset words. The tokenized data shape is a matrix, a rectangular array of research dataset files and the tokens in them, arranged in rows and columns. The shape of the research dataset after tokenization is (11,110, 1,742), with random word masking augmentation. With random token insertion augmentation, the shape of the dataset is (7,373, 1,885).

This represents an array of dataset files and their features. The dataset labels are converted into an array to be in the same format as the encoded dataset. The dataset and the label are stored in *X* and *Y* variables respectively. The implementation in Python is in Appendix C.1.5.

Algorithm 5 *Tokenizer* Feature Extraction Algorithm

1. Get *Tokenizer* function from the Keras library and store to a *tokenizer* variable
 2. Fit the *tokenizer* variable to convert words into numbers (features)
 3. Return the fitted *tokenizer*
 4. Use the *create_tokenizer* function to fit the *tokenizer* variable to the research dataset (set of augmented data from 5.2.3)
 5. Use the *tokenizer* to convert the list of words in augmented data into a matrix of numbers (features), using frequency count
 6. Save *tokenizer* as a pickle file for future usage
-

***TF-IDF* Feature Extraction**

The resulting TF-IDF data frame from the process in Algorithm 6 provides a representation of the dataset into features. Each row represents a file and the column represents a feature (term). The cells in the data frame represent the TF-IDF scores for each term in the corresponding dataset file. TF-IDF converts the text data into a numerical representation that encapsulates the importance of the words in each file relative to the entire dataset. After TF-IDF vectorization, the shape of the research dataset with random word masking augmentation is (11,110, 1,639). With random token insertion augmentation, the shape of the dataset is (7,373, 1,893). The TF-IDF Python implementation is shown in Appendix C.1.6.

Algorithm 6 *TF-IDF* Feature Extraction Algorithm

1. Import the *TfidfVectorizer* module from Sklearn library
 2. Create an instance of *TfidfVectorizer* and store to a *tfidf* variable
 3. Call the *fit.transform* method to fit the vectorizer on the research dataset (set of augmented data from 5.2.3) and transform it to TF-IDF features.
 4. The resulting TF-IDF matrix is stored in the variable *tfidf_data*
 5. The *toarray* method is called to convert *tfidf_data* into a matrix representation
 6. Call the *get_feature_names* method on *tfidf* to retrieve the feature names that correspond to the columns of the TF-IDF matrix
 7. Save *tfidf* vectorizer as a pickle file for future usage
-

5.2.5 Data Splitting

For machine learning algorithms to learn SQL query patterns in the dataset, the algorithms must be trained. To begin training, the research dataset is split into the training set and the testing set. The training set is the amount of the dataset exposed to the machine learning algorithm to study during training while the testing set is used to subsequently measure the performance of the algorithm. This works by passing the dataset for testing into the machine learning model, for the model to make a prediction (classification). The prediction is then compared to the actual values in the testing set.

To implement this in Python, the *sklearn* library is imported. Sklearn (scikit-learn) is a Python library used for the implementation of various machine learning algorithms. Before the implementation of algorithms to train a model is commenced, splitting the research dataset is required. *Sklearn* learning has a function to split datasets into training and testing sets and this is imported into the project for usage. The dataset is divided into four sets; *input training set*, *input testing set*, *output training set*, and *output testing set*. When training machine learning models, the input training set and the output training set are passed into the models. For the MLP classifier training, 33% of the research dataset is used for model testing. 20% of the dataset is reserved for testing in the NB, KNN, and LR classifiers. This value is a hyperparameter and it is subject to tuning.

As mentioned in 5.2.4, the shape of the research input training data is (11,110, 1,742) with random word masking and *Tokenizer* feature selection, (7,373, 1,885) with random token insertion and *Tokenizer*, the shape is (11,110, 1,639) with random word masking and TF-IDF and it is (7,373, 1,893) with random token insertion and TF-IDF.

In the MLP classifier, the training set is made of 67% of the research dataset. Therefore, with random word masking, 7,443 of the SQL query samples are used in model training and 3,667 samples are reserved for testing the model performance. 4,939 samples are used for training and 2,434 are used for model testing when the dataset is augmented using random word insertion. 80% of the dataset is used in training the NB, KNN, and LR classifiers. With random word masking, this is made up of 8,888 SQL query samples, and the remaining 2,222 samples are used

for testing and evaluating the models. Random word insertion has 5,898 training samples and 1,474 samples are used for testing the model performance after training.

Algorithm 7 Data Splitting Algorithm

1. Import function to split the dataset into training and testing set from *sklearn*
 2. Call function from step 1 and pass four arguments; variable for tokenized dataset (X), variable for dataset labels (Y), amount of dataset to use for testing
 3. Store step 2 into four variables that represents the input training, input testing, output training and output testing data
-

5.3 Re-identification Query Pattern Recognition Using Binary Classification Algorithms with *Tokenizer* Feature Extraction

This section presents the design and training of the binary classification models implemented for the recognition of patterns in SQL queries that suggest data re-identification attack attempts on anonymised databases. Also, the exploration of essential parameters to improve training in the classifiers is presented. For this research dataset, four classification algorithms are explored; Multilayer Perceptron (neural network), Naive Bayes, K-nearest Neighbors, and Logistic Regression. Designing and training the MLP model requires setting hyperparameters; including the number of layers and nodes in the neural network, activation functions, number of training iterations, and loss function based on the nature of the classification problem. The approach for determining the optimum hyperparameters to use for training the NB and LR classifiers involved creating an initial model that utilises the default hyperparameter values from the scikit-learn library, then training a tuned version of the models to improve performance. Optimum hyperparameter values are explored using grid search cross-validation to create and test multiple models with the combination of different hyperparameters to determine the hyperparameter values best suited for the classifiers to efficiently learn the intricacies in the research dataset. The initial and the tuned model are both evaluated to compare performance on the test dataset. Training the KNN classifier involved using grid search CV (cross-validation) to determine an ideal value of k . The k value is then used to explore the KNN algorithm with both Euclidean and Manhattan distance metrics to generate two KNN models. A performance evaluation of both KNN models is presented in 5.3.3.

The resulting datasets from both the random word masking and random token insertion data augmentation techniques are used in the machine learning experiment presented in this work. This section explores the training of different machine learning algorithms using the two sets of augmented data and the *Tokenizer* feature extraction technique. Across all the classifiers implemented in this research, the experiment using random word masking augmented dataset showed better training results than the random token insertion dataset. This could be a result of the random word masking augmentation method generating a higher number of augmented datasets (SQL query samples) than random token insertion. This is also an indication that the random word masking augmentation technique is better suited for the research dataset, as it generated more data samples and performed better during training with classification algorithms.

As such, this section only presents the training results for the research experiment using the dataset generated from random word masking. The training result from experimenting with random token insertion is presented in Appendix D.2.

5.3.1 Multilayer Perceptron Classifier

As discussed in 3.8.1, the neural network architecture implemented in this research utilises the dense layer from *keras*, with a fully connected layer. To train a fully connected Multilayer Perceptron classifier, two functions from the *keras* library are required; the *Sequential* and the *Dense* functions. The *Sequential* API is based on sequencing multiple NN layers, it is a linear pipeline (a stack) of NN layers. The *Sequential* function is imported to build a neural network with multiple layers and the *Dense* function is specified to create fully connected layers in the neural network architecture.

The topology of the multilayer perceptron neural network designed for the classification of re-identification attack attempts consists of three layers (a simple neural network architecture to training the research dataset), as discussed in section 3.9.1, a complex model tends to overfit the dataset. The input layer comprising of the input data and bias, the input data is the 1,742 numerical features extracted from the dataset for random word masking and 1,885 for random token insertion, as detailed in 5.2.4. An additional bias value of 1 is added at the input layer, making the total inputs at this layer 1,743 and 1,886 with random word masking and random token insertion respectively. As mentioned in the discussion in section 3.8.1, the bias input value is traditionally set to 1 when implementing an MLP [21, 56]. At the hidden layer, 500 neurons are used. Another bias value is added at this layer making the total input at the hidden layer 501. The output layer consists of a single neuron, for the computation of the output. The output layer in this architecture requires a single node due to the nature of the classification task being binary. A single output value is expected in binary classification to signify the predicted class label. The *ReLU* and *sigmoid* activation functions are used at the hidden and output layers respectively. *Binary cross-entropy* loss function is used to minimize the errors in the neural network during training and the *Adam* optimizer is selected to control the learning rate of the model during training. The justification for the neural network parameters used is covered in section 3.8.1.

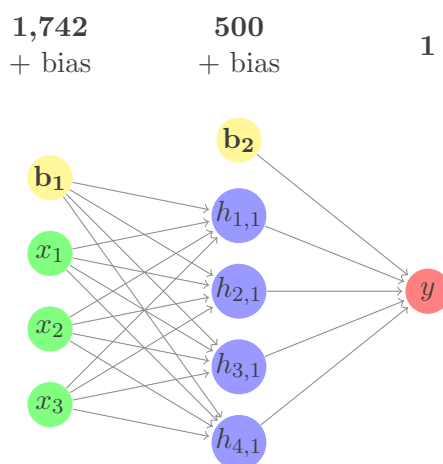


Figure 5.1: MLP classifier for re-identification attempt recognition (random word masking)

The creation of a neural network model is defined by the computation of the specified hyper-

parameters (number of layers and neurons). Using the random word masking augmented dataset, the NN in this work has $(1743 \times 500) + (501 \times 1)$ parameters in its design. This equates to 872,001 trainable parameters in the neural network model. Appendix D.1.1 shows the model parameters for this neural network design. With random token insertion, the NN has $(1886 \times 500) + (501 \times 1)$ parameters. This equates to 943,501 trainable parameters in the model.

After the design and definition of the MLP model as shown in Figure 5.1 and Appendix D.1.1 respectively, the model is trained with the training dataset. This consists of the input training set (features for 7,443/4,939 query samples) and the output training set (labels for 7,443/4,939 query samples). The training process in this neural network model is repeated a specified number of times known as *epoch* (iterations), this is a hyperparameter that states the number of times that the learning algorithm is exposed to the entire dataset. During training, the MLP for this work uses backpropagation to report the loss (error) in the model and update the weights. Mini-batch gradient descent is used and batch size is set at 32. The batch size hyperparameter specifies the number of samples the model studies before updating the model weights. For every batch, predicted labels (outputs) are compared to the expected (actual) labels, to calculate a loss value. The mini-batch gradient descent optimization algorithm then updates the model weights to improve its performance. The calculation of error during training of the neural network model is done using 20% of the training dataset for *validation*. The validation split is used by the model to cross-check and adjust how well the model is learning the principles of the dataset. The validation dataset is not included in the model training but is used for model loss and accuracy evaluation.

With the random word masking augmented dataset, the 20% validation split equates to 1,489 samples, therefore, 5,954 samples are used for model training. This equates to 3,951 samples being used for model training and 988 samples used as the validation split with random token insertion. With this amount of training data, an epoch of 10, and a batch size of 32, the dataset is divided into approximately 187 batches ($5,954 \div 32$) and approximately 124 batches ($3,951 \div 32$) for the two augmented datasets. The model weights are updated after every batch of 32 samples. This is repeated for 10 epochs to optimize and improve model performance, with 187 and 124 batches per epoch and a total of 1,870 and 1,240 batches during the overall training process. The training process for this neural network model using both datasets is in Appendices D.1.2 and D.2. Over the 10 iterations, the training and validation accuracy and loss are presented in Figures 5.2 and 5.5 below, for random word masking and random token insertion datasets respectively.

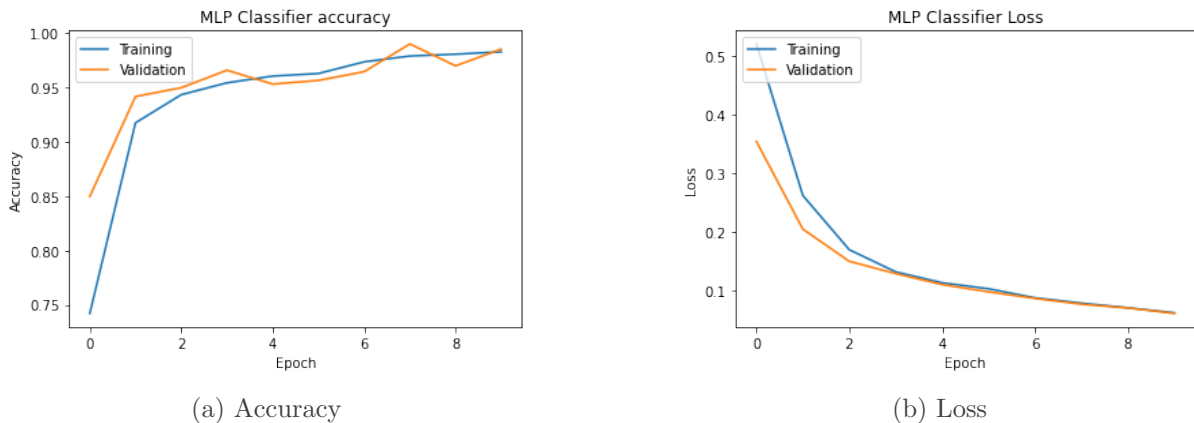


Figure 5.2: MLP classifier training accuracy and loss over 10 epochs (random word masking)

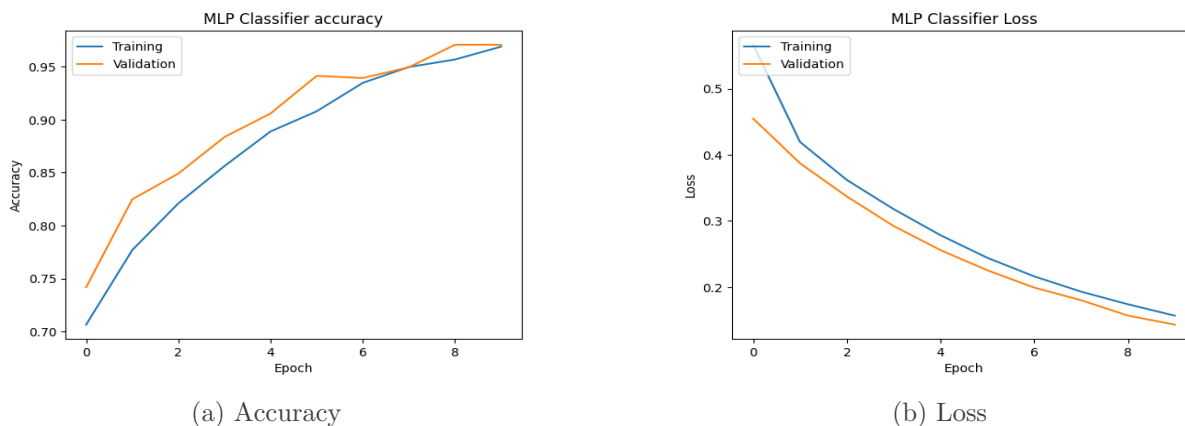


Figure 5.3: MLP classifier training accuracy and loss over 10 epochs (random token insertion)

At the end of 10 epochs, the MLP model achieved 98.3% (0.983) accuracy and 98.5% (0.985) validation accuracy during training using the random word masking augmented dataset. The model achieved 96.9% accuracy and 97.1% validation accuracy with the random word insertion dataset. This suggests a better performance with the random word masking data augmentation technique. The training results show that the model was successful in learning the principle in the training dataset, indicated by the high accuracy percentage score. A significant discrepancy in the training accuracy and validation accuracy results will suggest high training errors in the model. Using random word masking, 98.7% testing accuracy was achieved by the MLP classifier and 96.8% with random word insertion after its performance is evaluated with the 33% test data as presented in 6.1.1 and Appendix D.15. The trained MLP classifier model and the model weights are saved to the file for usage.

5.3.2 Naive Bayes Classifier

Four Naive Bayes classifiers from scikit-learn (Gaussian, Bernoulli, Multinomial, and Categorical) are fitted to the training dataset. To generate a baseline model, the different NB classifiers are trained using the default hyperparameter arguments recommended in scikit-learn. Using the accuracy metric, the performances of the classifiers are compared to determine which is most suitable for the classification of the research dataset. This performance comparison is detailed in chapter 6. The Gaussian NB classifier achieves an accuracy of 0.778 (77.8%), Bernoulli NB has an accuracy score of 0.793 (79.3%), the Multinomial NB classifier returned an accuracy score of 0.838 (83.8%) and the Categorical NB achieves an accuracy of 0.495 (49.5%). The multinomial naive Bayes classifier returned the best performance metrics among the different Naive Bayes classifiers. As mentioned in 3.8.2, this can be attributed to the fact that multinomial NB classifiers are most suitable for classification problems involving textual data that uses word frequency count in its classification operation, as in the approach for this research data mining experiment. The default hyperparameter-trained multinomial NB is selected as the baseline classifier.

To generate a tuned version of the multinomial NB classifier, the *alpha* value hyperparameter is tuned with grid search CV. This hyperparameter is responsible for smoothing in the NB algorithm. This value is set to 1 by default and that is used in the baseline model. To explore a more optimal *alpha* value for smoothing, a grid search CV function is employed. This is detailed in Algorithm 8 below.

Algorithm 8 Multinomial Naive Bayes Grid Search Cross-Validation

1. Import *MultinomialNB* classifier from scikit-learn
 2. Import *GridSearchCV* from scikit-learn
 3. Initialise *MultinomialNB* classifier
 4. Set *alpha* hyperparameter to try values between -2 and 2 on the logarithm scale
 5. Pass the initialised model, 5-fold CV, and the hyperparameter dictionary into a *GridSearchCV* function
 6. Fit the model to the training dataset
 7. Return the *MultinomialNB* classifier *alpha* hyperparameter value with the best fit
 8. Train *MultinomialNB* with the returned value
 9. Save the classifier model to file
-

Grid search CV returned $\alpha = 0.01$ as the optimum value for training a multinomial NB classifier on the research dataset. After training with the tuned hyperparameter, the classifier achieved an accuracy score of 0.844 (84.4%). A detailed performance evaluation of the baseline and tuned classifier is presented in 6.1.2.

5.3.3 K-Nearest Neighbours Classifier

As discussed in section 3.8.3, KNN is a lazy learner that simply memorises the dataset and performs the required estimations during the classification of a new sample. The essential hyperparameter to explore is the k value and the distance metric to be used for the estimation of similarities in sample features. Using grid search CV, an optimal value of k is explored for the research training dataset. Grid search CV is set to use 5-fold cross-validation and to explore odd integers between 1 and 30, as mentioned in section 3.8.3, odd integers are used to avoid a tie in the numbers of neighbors used when deciding class membership for a new sample. Algorithm 9 below presents the search for the optimum k value to set for KNN training.

Algorithm 9 K-Nearest Neighbors Grid Search Cross-Validation

1. Import *KNeighborsClassifier* algorithm from scikit-learn
 2. Import *GridSearchCV* from scikit-learn
 3. Initialise *KNeighborsClassifier* algorithm
 4. Set *n_neighbors* (*k* value) hyperparameter to try values within the 1 and 30 range, with an increment of 2 to keep the (*k* value) odd
 5. Pass the initialised model, 5-fold CV, and the hyperparameter dictionary into a *GridSearchCV* function
 6. Fit the model to the training dataset
 7. Return the *k* hyperparameter value with the best fit
 8. Train KNN with the returned value
 9. Save model to file
-

Grid search CV returned the $k = 5$ as the value best suited to train a KNN model for the research training dataset. k value of 5 is also recommended by scikit-learn as the default number of neighbors to consider when KNN is estimating feature similarities. A KNN model is then trained with a k of 5 and *Euclidean distance*. The model achieved an accuracy of 99.9%. Another KNN model with the same k value and the *Manhattan distance* metric is trained, with a resulting accuracy score of 100%. The performance evaluation for the baseline model with default hyperparameters and the tuned model is detailed in 6.1.3.

5.3.4 Logistic Regression Classifier

From the scikit-learn library in Python, the logistic regression function is imported to begin the training of the classifier. Following the same strategy as in 5.3.2 and 5.3.3, a baseline model is created to evaluate how well logistic regression fits the research dataset. The baseline model uses the default logistic regression hyperparameter values from scikit-learn. This has the C value set to 1.0 and uses the *lbfgs* solver for optimization. The LR baseline model achieved an accuracy score of 0.915 (91.5%). Grid search cross-validation function is implemented to improve the model performance, using 5-fold cross-validation. First, grid search CV is used to determine if there are differences in the performance of logistic regression based on solvers. The five available solvers are tested against the dataset to return the *mean test score* for each of the solvers. This divides the dataset into 5 folds and trains the LR classifier with $k-1$ (4) folds of the dataset and tests with a different fold for 5 iterations. This is done for all the solvers. The *mean test score* for the accuracy achieved by each solver shows that the solvers have similar performance rates on the dataset. The returned scores shows *liblinear* at 0.887, *newton-cg* at 0.887, *lbfgs* at 0.887, *sag* at 0.886 and *saga* at 0.887. This suggests that the 5 solvers have a similar ability in optimizing the error in the logistic regression classifier algorithm.

Grid search CV function is then used for an exhaustive search to determine an optimum combination of hyperparameter values to set for logistic regression in an attempt to improve the

performance quality of the model. A dictionary variable is declared to store all the hyperparameter values to be tested for the model creation. Algorithm 10 depicts the process of implementing the hyperparameter search for logistic regression in Python.

Algorithm 10 Logistic Regression Grid Search Cross-Validation

1. Import *LogisticRegression* classifier from scikit-learn
 2. Import *GridSearchCV* from scikit-learn
 3. Initialise *LogisticRegression* classifier
 4. Set *C* hyperparameter to try values between -4 and 4 on the logarithm scale
 5. Set *solver* hyperparameter to try all the 5 logistic regression solvers
 6. Pass the initialised model, 5-fold CV, and the hyperparameter dictionary into a *GridSearchCV* function
 7. Fit the model to the training dataset
 8. Return the logistic regression model hyperparameter values with the best fit
 9. Train logistic regression model with the returned values
 10. Save model to file
-

The returned model from grid search CV implements the *Newton-cg* solver and a *C* value of 0.0001. The returned *C* value is consistent with the point that, the smaller the value of *C* the better the regularization in logistic regression models, as discussed in section 3.8.4. The newly derived hyperparameters are passed to train a new logistic regression model on the research dataset. The new model achieved an accuracy score of 1.0 (100%). Both the baseline and the tuned models are evaluated with the 20% test data, the evaluation of the classification performance by the two models is presented in 6.1.4.

5.4 Re-identification Query Pattern Recognition Using Binary Classification Algorithms with *TF-IDF* Feature Extraction

As discussed in section 5.2.4, the TF-IDF feature extraction technique is also implemented in the machine learning experiment for this research. The experimentation with TF-IDF uses the same classification algorithms and parameters as in section 5.3 for the design and training of a new set of classifiers. Random word masking data augmentation appears to be a more suitable augmentation technique for the research dataset when compared to the random token insertion technique. This can be implied from the fact that random word masking generated more data samples, as highlighted in section 5.2.3. Also, there is a better overall performance recorded in the classifiers trained using the random word masking augmented dataset. Therefore, the training of TF-IDF classification models is executed using the dataset generated with random word masking augmentation.

5.4.1 Multilayer Perceptron Classifier

The TF-IDF neural network architecture followed the same structure as the one implemented in section 5.3.1. However, as mentioned in 5.2.4, the shape of the research dataset with random work masking augmentation and TF-IDF feature extraction is (11,110 1,639), indicating 11,110 samples and 1,639 extracted numerical features. This means that the input layer of the neural network in this case is comprised of 1,639 features and an additional bias value of 1, equaling a total of 1,640 input features. 500 neurons are used at the hidden layer, and a bias value is also added at this layer, adding up to 501 features. The architecture of the TF-IDF multilayer perceptron is depicted in figure 5.4 below.

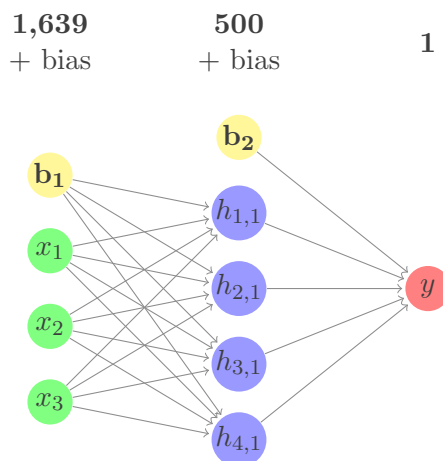


Figure 5.4: TF-IDF MLP classifier for re-identification attempt recognition

The TF-IDF neural network has $(1640 \times 500) + (501 \times 1)$, equating to 820,501 trainable parameters in the neural network model. With 33% of the dataset reserved for testing, the model is trained with 7,443 query samples, 10 epochs, a batch size of 32, and a 20% validation split. The training and validation accuracy and error are presented below in figures 5.5a and 5.5b respectively.

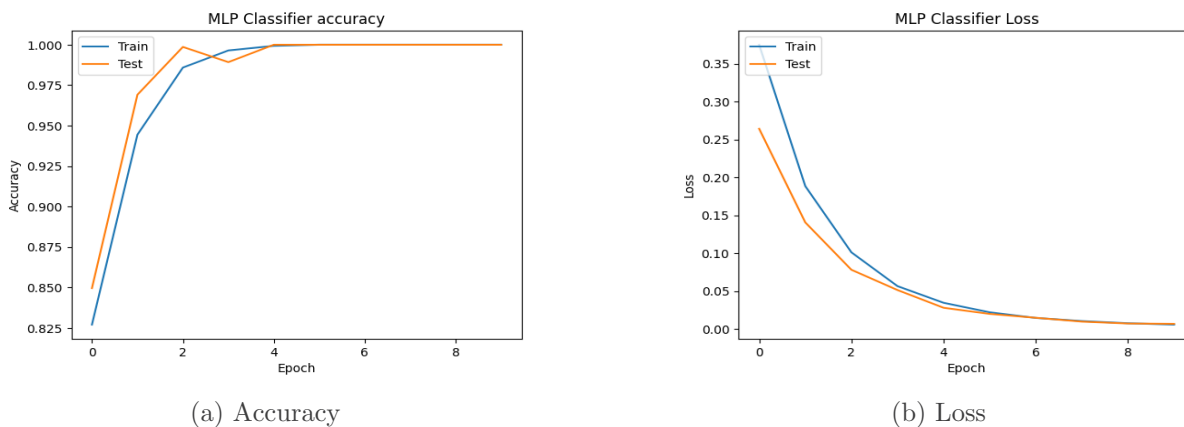


Figure 5.5: TF-IDF MLP classifier training accuracy and loss over 10 epochs

The TF-IDF MLP model achieved an accuracy of 100% (1.0) and a validation accuracy of 100% (1.0) over 10 epochs. After testing with the 33% test dataset, the model achieved a testing accuracy of 100%. The performance evaluation for the TF-IDF MLP model indicates better training and testing results in comparison to the MLP trained using the *Tokenizer* feature extraction.

5.4.2 Naive Bayes Classifier

Similar to the Naive Bayes classifiers trained in section 5.3.2, four NB classifiers are trained using TF-IDF features. Adopting the same strategy, the classifiers are initially trained with default hyperparameters to deduce a baseline performance. The performances are then compared among the NB classifiers. Using TF-IDF features, the Gaussian NB classifier achieves an accuracy of 0.779 (77.9%), Bernoulli NB has an accuracy score of 0.781 (78.1%), the Multinomial NB classifier returned an accuracy score of 0.783 (78.3%) and the Categorical NB achieves an accuracy of 0.495 (49.5%). These indicate similar performance to the NB classifiers trained with *Tokenizer* features. Also, the Multinomial NB returned the best accuracy score among the implemented NB models.

A tuned Multinomial NB classifier is subsequently trained to explore a better-performing NB classifier for the research dataset. The tuned Multinomial NB model returned an accuracy of 0.793 (79.3%). This is slightly better than the accuracy recorded for the baseline Multinomial NB, however, the performance accuracy score with the TF-IDF features is inferior to that achieved with the *Tokenizer* features.

5.4.3 K-Nearest Neighbours Classifier

Using the same process detailed in algorithm 9, two KNN classifiers are trained using TF-IDF features. A KNN classifier is trained with a k value of 5 and the *Euclidean distance*, and another KNN model is trained with the *Manhattan distance* metric. The *Euclidean distance* trained model returned an accuracy score of 0.999 (99.9%) and the *Manhattan distance* trained model returned a 1.0 (100%) accuracy score. These scores are consistent with those returned in the KNN models trained with *Tokenizer* features. This consistency in the KNN scores at this point can be attributed to the KNN being a lazy learner and simply storing the dataset in memory.

5.4.4 Logistic Regression Classifier

With the TF-IDF dataset features, both a baseline model with default hyperparameters and a tuned model with hyperparameters generated in section 5.3.4 are trained. The training adopts the same process and strategy implemented while training with the *Tokenizer* features. The baseline logistic regression model achieved an accuracy of 0.908 (90.8%), a slightly inferior score to the baseline LR model with *Tokenizer* features. A tuned version of the model is trained using the *Newton-cg* solver, and a C value of 0.0001. After testing the tuned LR model achieved an accuracy of 1.0 (100%), matching the performance of the tuned LR model previously trained with *Tokenizer* features.

5.5 Chapter Summary

This chapter presents the data mining experiment conducted for this research. Details about the SQL database queries gathered for the research are explored. The approach implemented for the preprocessing, labelling, tokenization, augmentation, and feature extraction of the SQL query data is presented in this chapter. The data augmentation process explored, compared, and experimented with two different techniques. The techniques are random word masking and random token insertion augmentation. Also, a comparative approach was adopted for the feature selection process, with the experiment exploring both the *Tokenizer* and the *TF-IDF* feature extraction methods for the vectorization of the research dataset. The different Python libraries and techniques used at each stage of the data mining experiment are described in this chapter. The dataset is labeled for a supervised learning experiment, label 0 is appended to the normal SQL query sample and 1 to the samples where re-identification is intended. Due to the limited query sample size in the data collection for this research, this chapter detailed the use of data augmentation techniques used in generating more data from the gathered dataset in section 4.6. This chapter outlines the process of encoding the words in the research dataset into numerical values that can be processed by machine learning algorithms (feature extraction).

The implementation of different binary classification algorithms is presented in sections 5.3 and 5.4. Two MLP classifiers (using the two different feature extraction methods) are trained for the classification problem in this work using 67% of the dataset for training and 33% for testing in the MLP model. The MLP classifiers achieved an accuracy of 98.7% and 100% with *Tokenizer* and *TF-IDF* feature extraction respectively when evaluated with the test set. For the NB, KNN, and LR classifiers, 20% of the dataset is reserved for testing, and 80% is used to train these different algorithms. The strategy employed with the NB and LR classifiers involved creating an initial baseline model and subsequently optimizing the hyperparameters in the models to generate a tuned version of the classifier. With *Tokenizer* features, the baseline and the tuned Multinomial NB models achieved an accuracy of 83.8% and 84.4% respectively. The KNN model implementing Euclidean distance achieved an accuracy of 99.9% while the KNN Manhattan distance metric achieved a 100% accuracy. The accuracy score increased from 91.5% in the baseline model to 100% in the tuned model in the case of logistic regression. Using TF-IDF, the baseline and the tuned Multinomial NB models achieved an accuracy of 78.3% and 79.3% respectively. The KNN model implementing Euclidean distance achieved an accuracy of 99.9% with the KNN Manhattan distance metric achieved a 100% accuracy. Logistic regression achieved 90.8% accuracy with the baseline model and 100% with the tuned model. The tuned models are generated from an exhaustive hyperparameter search using the grid search CV function from scikit-learn. The models trained in this chapter are evaluated and compared to determine which models are most effective for the recognition of re-identification query patterns. The details of the evaluation and comparison of performances in these models are presented in chapter 6.

Chapter 6

Re-identification Query Pattern Recognition Models: Performance Evaluation and Analysis

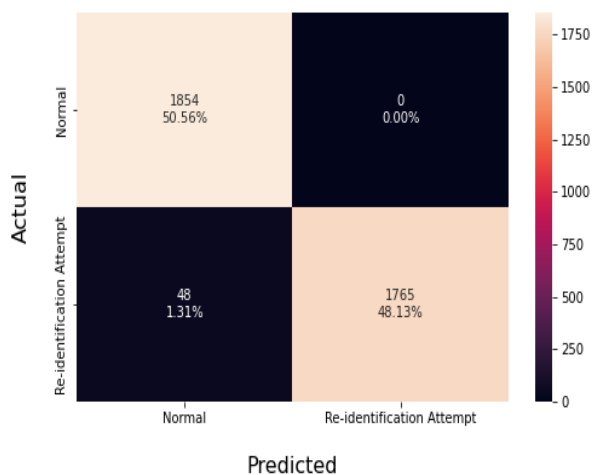
The data mining experiment described in chapter 5 presents the training process of four classification algorithms for the recognition of patterns in SQL queries that may lead to the re-identification of subjects of an anonymised database. In this chapter, the performance evaluation of the different models on the testing dataset is presented and analysed. Section 6.1 focuses on the discussion around the classification and misclassification rates of the models when evaluated with the test data. The discussion presents a comparative analysis of the model performances when trained with the two different feature extraction techniques. A comparison in the performance of the applied models is detailed in section 6.2. Sections 6.4 and 6.5 present the testing of models with the unseen dataset and analysis of the test results respectively.

6.1 Model Performance Evaluation

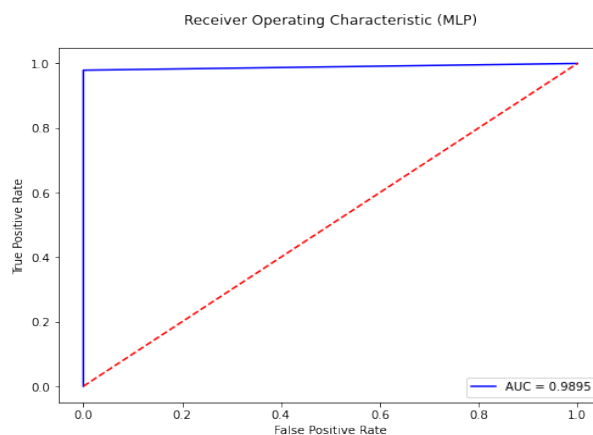
Section 3.9 presents discussions around the use of the confusion matrix as a tool for evaluating the classification performance of a predictive model. The performance evaluation of models in this work uses confusion matrices to analysis the percentage rate of accurate classification and misclassification of both the normal class and the re-identification attack attempt class. The classification experiment in this work developed baseline models and tuned models for the NB, KNN, and LR classifiers, using both the *Tokenizer* and the *TF-IDF* feature extraction datasets. The performance of models is evaluated and analysed. A confusion matrix is used to evaluate and compare performances in baseline and tuned versions of the models. Threshold-based metrics such as *accuracy*, *precision*, *F-score* and *recall* are used in the evaluation of model performances. A rank-based metric such as AUC is also used to evaluate and compare the models. As previously discussed, 33% of that research dataset is reserved for testing in the MLP models. This is 3,667 of the query samples, 1,854 of which are of the normal class and 1,813 belongs to re-identification attempt class. 20% (2,222 samples) test set is used in the Naive Bayes, k-nearest neighbors, and logistic regression models. This includes 1,123 normal samples and 1,099 re-identification attempt samples.

6.1.1 Evaluation of Re-identification Query classification with Multilayer Perceptron

The confusion matrices for the performance evaluation of both the *Tokenizer* and the *TF-IDF* MLP models with the testing dataset are presented in 6.1a and 6.2a below. ROC curves calculating the AUC score for the models are shown in Figures 6.1b and 6.2b.

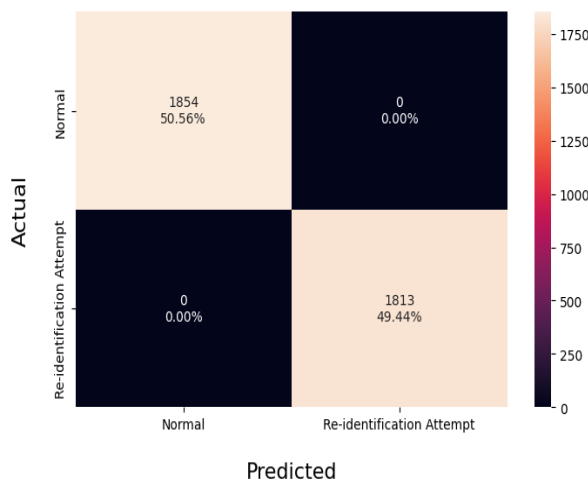


(a) MLP confusion matrix

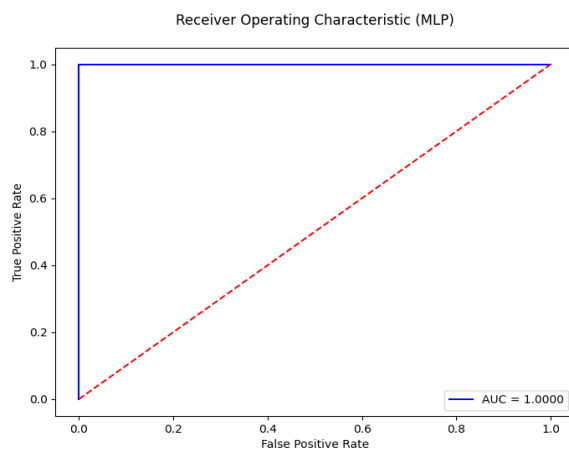


(b) ROC curve for MLP classifier

Figure 6.1: *Tokenizer* MLP classification performance evaluation confusion matrix and ROC curve



(a) MLP confusion matrix



(b) ROC curve for MLP classifier

Figure 6.2: *TF-IDF* MLP classification performance evaluation confusion matrix and ROC curve

The confusion matrix in Figure 6.1a shows that the *Tokenizer* MLP model correctly classified 48.13% (1,765 query samples) of the testing dataset as re-identification attempts and 50.56% (1,854 query samples) of the test set are correctly recognised as normal queries. The misclassification in the model shows 1.31% (48 query samples) of re-identification attempts queries incorrectly predicted as normal. None of the query samples labelled as normal are misclassified

as re-identification attempts, showing a 0% false positive in the model evaluation. The MLP score classification performance evaluation resulted in a 98.7% score for accuracy, precision, recall, f-score, and AUC. A full classification report for the MLP model testing is in Appendix D.1.3.

Figure 6.2a depicts the classification performance evaluation for TF-IDF with a 100% accuracy score. This shows that the model correctly classified all re-identification attempt test samples as attack attempts and all normal query samples are classified as such. The model returned 0% false positive and false negative rates and an AUC score of 100%. A full classification report for the model is in Appendix D.3.3.

Classifiers	Accuracy (%)	Precision (%)	Recall (%)	F-score (%)	ERR (%)	TNR (%)	FPR (%)	AUC (%)
<i>Tokenizer</i> MLP	98.7	98.7	98.7	98.7	1.3	100	0	98.7
<i>TF-IDF</i> MLP	100	100	100	100	0	100	0	100

Table 6.1: Comparison of performance evaluation metrics in different MLP models

6.1.2 Evaluation of Re-identification Query classification with Naive Bayes

The experiment discussed in section 5.3.2 and 5.4.2 implements four Naive Bayes classifiers with default parameters to generate baseline models, using the *Tokenizer* and *TF-IDF* features respectively. The Gaussian NB, Bernoulli NB, Categorical NB, and Multinomial NB are explored. The models that fit the research dataset best are selected to be further explored with hyperparameters tuning. Multinomial NB performs best with both feature extraction techniques, based on the evaluation of results from the classification of the testing dataset. The evaluation of the classification performance achieved by the four baseline NB models with the two different feature extraction is represented by the confusion matrices in Figures 6.3 and 6.4.

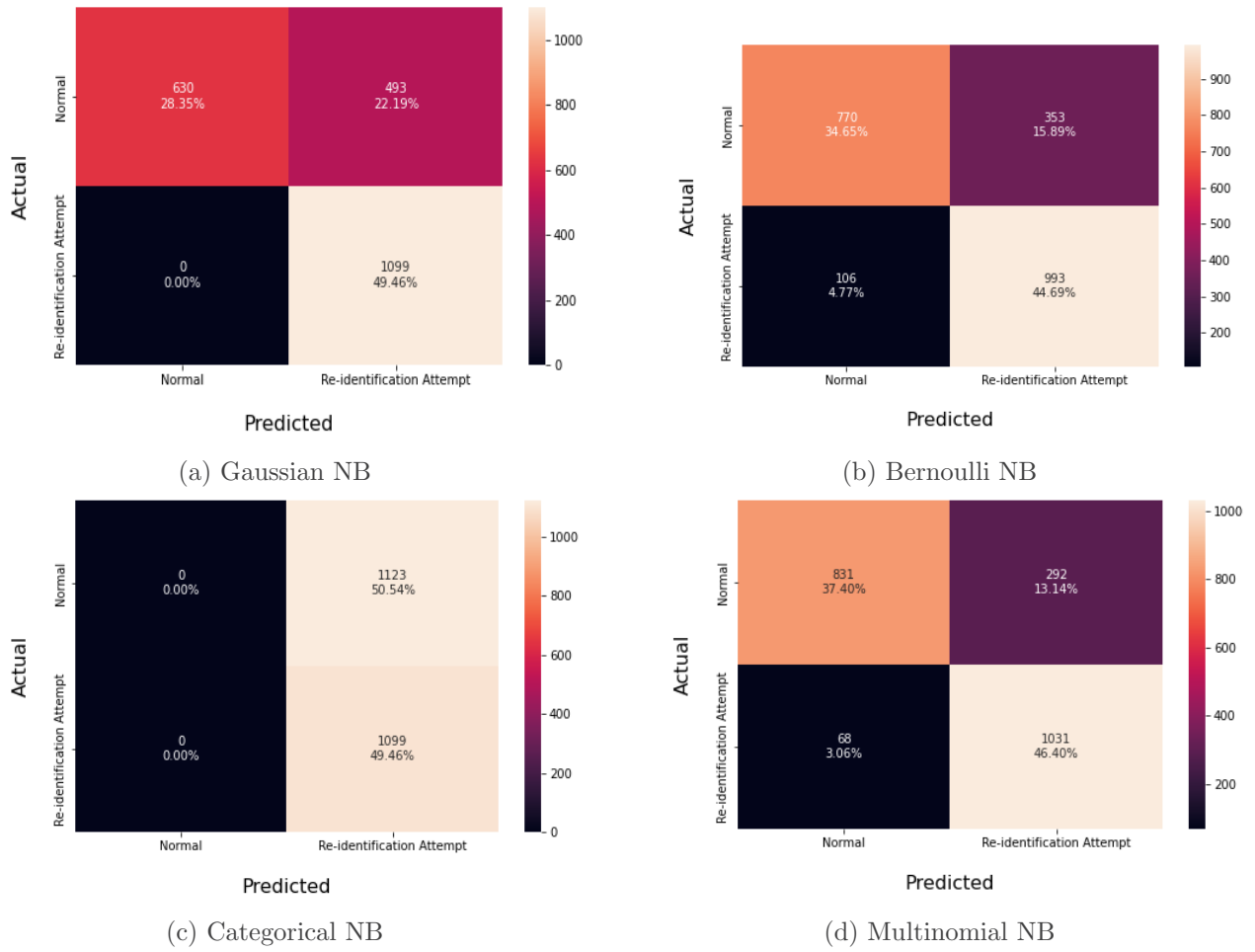


Figure 6.3: Classification performance evaluation confusion matrices for different NB classifiers with *Tokenizer*

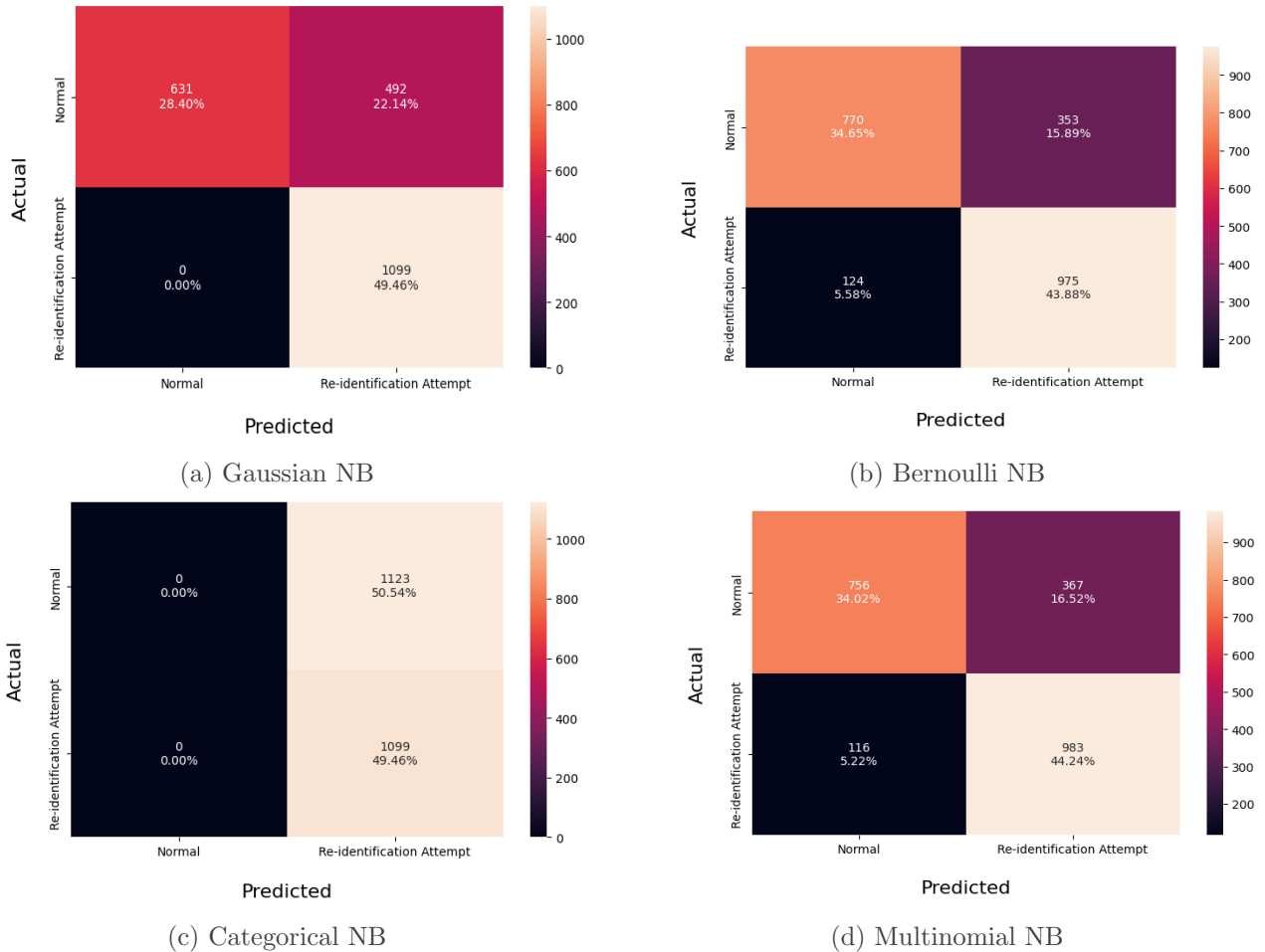


Figure 6.4: Classification performance evaluation confusion matrices for different NB classifiers with *TF-IDF*

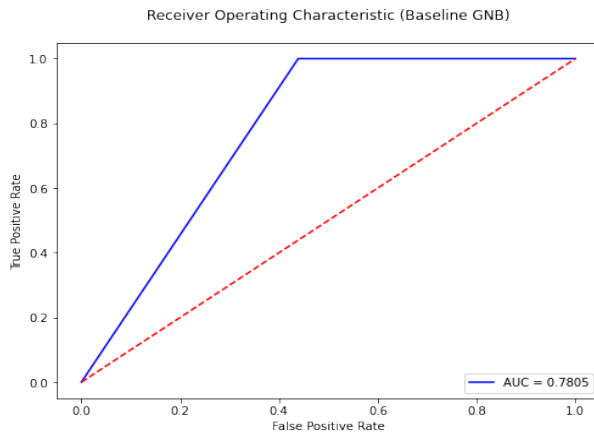
Using the *Tokenizer* features, figure 6.3a shows that the Gaussian NB model correctly predicts 1,099 re-identification query samples as re-identification attempts. 630 normal query samples are correctly classified (true negatives) and the remaining normal samples are misclassified by the model, incorrectly predicting 493 normal samples as re-identification attempts. This suggests a significant false positive rate. There are 0% false negatives in the model, with no re-identification sample misclassified as normal. The performance evaluation results are almost exactly the same for the GNB using the *TF-IDF* features, with the differences being one more true negative sample and one less false positive sample. The Gaussian NB with *Tokenizer* reported a 77.8% accuracy on the test set, 84.7% for precision, a recall of 77.8%, an f-score of 76.7%, and a 78.1% AUC score. With *TF-IDF*, the accuracy is 77.9%, 84.7% precision, recall of 77.9%, 76.8% f-score, and AUC is 78.1%. The classification reports for Gaussian NB models are in appendices D.1.4 and D.3.4.

As shown in Figure 6.3b, the Bernoulli NB classifier using the *Tokenizer* features correctly predicts 993 instances of the re-identification attempt samples and does the same for 770 of the normal query samples. 106 re-identification query instances are mistaken as normal and 353 normal instances are predicted incorrectly as re-identification attack attempts. There are 124 re-identification attempt query samples misclassified and 975 re-identification samples correctly

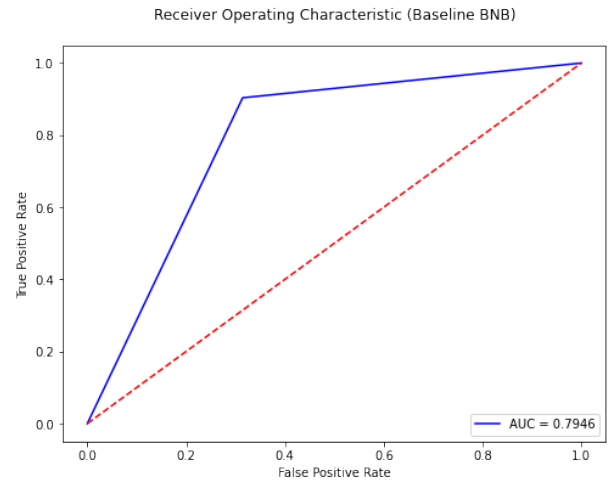
classified when using *TF-IDF*. The true negative and false positive rates with both feature extraction methods remain the same. The classification reports for the two Bernoulli NB are shown in appendices D.1.5 and D.3.5.

As represented with the confusion matrices in Figures 6.3c and 6.4c, Categorical NB correctly predicted all the 1,099 re-identification samples in the testing set. However, it misclassified all the 1,123 normal samples. There are no normal samples correctly predicted and no re-identification samples misclassified. The result is consistent in the Categorical NB, using both the *Tokenizer* and the *TF-IDF* features. The classifiers achieved an accuracy of 49.5%, a precision of 24.5% that indicates the classifier's high rate of negative as positive prediction, recall of 49.5% is achieved with 32.7% f-score and an AUC of 50%. Appendices D.1.6 and D.3.6 show the full classification performance reports for the Categorical NB classifiers.

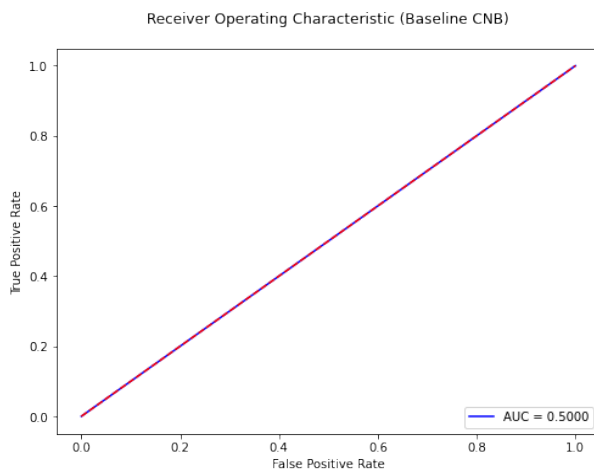
The Multinomial NB (*Tokenizer*) performance on the testing set as shown in Figure 6.3d, has 1,031 re-identification samples correctly classified. The true negative result shows 831 normal query instances correctly classified. The misclassification rate shows 68 re-identification attempt instances confused as normal and 292 normal samples confused as re-identification. The accuracy achieved in the Multinomial NB is 83.8%, a precision of 85.3%, recall of 83.8%, f-score of 83.7%, and an AUC of 83.9%. This model shows a better evaluation performance than the Multinomial NB trained using *TF-IDF* features with 983 re-identification attack attempt samples correctly classified, true negative results including 756 samples, 116 re-identification samples, and 367 normal samples misclassified. The accuracy achieved in the *TF-IDF* Multinomial NB is 78.3%, a precision of 79.8%, recall of 78.3%, f-score of 78.0%, and an AUC of 78.4%.



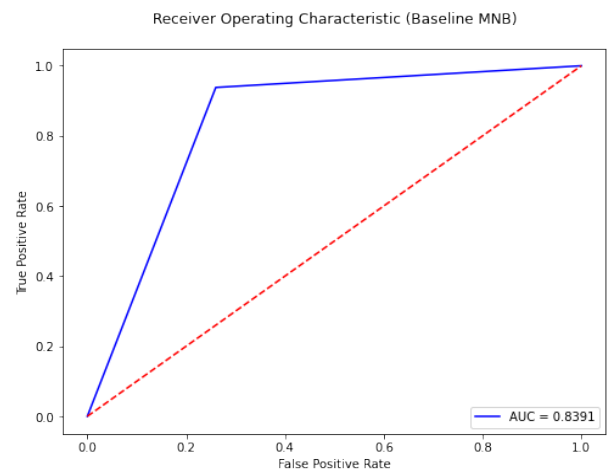
(a) Gaussian NB



(b) Bernoulli NB



(c) Categorical NB



(d) Multinomial NB

Figure 6.5: ROC curves for different NB classifiers using *Tokenizer*

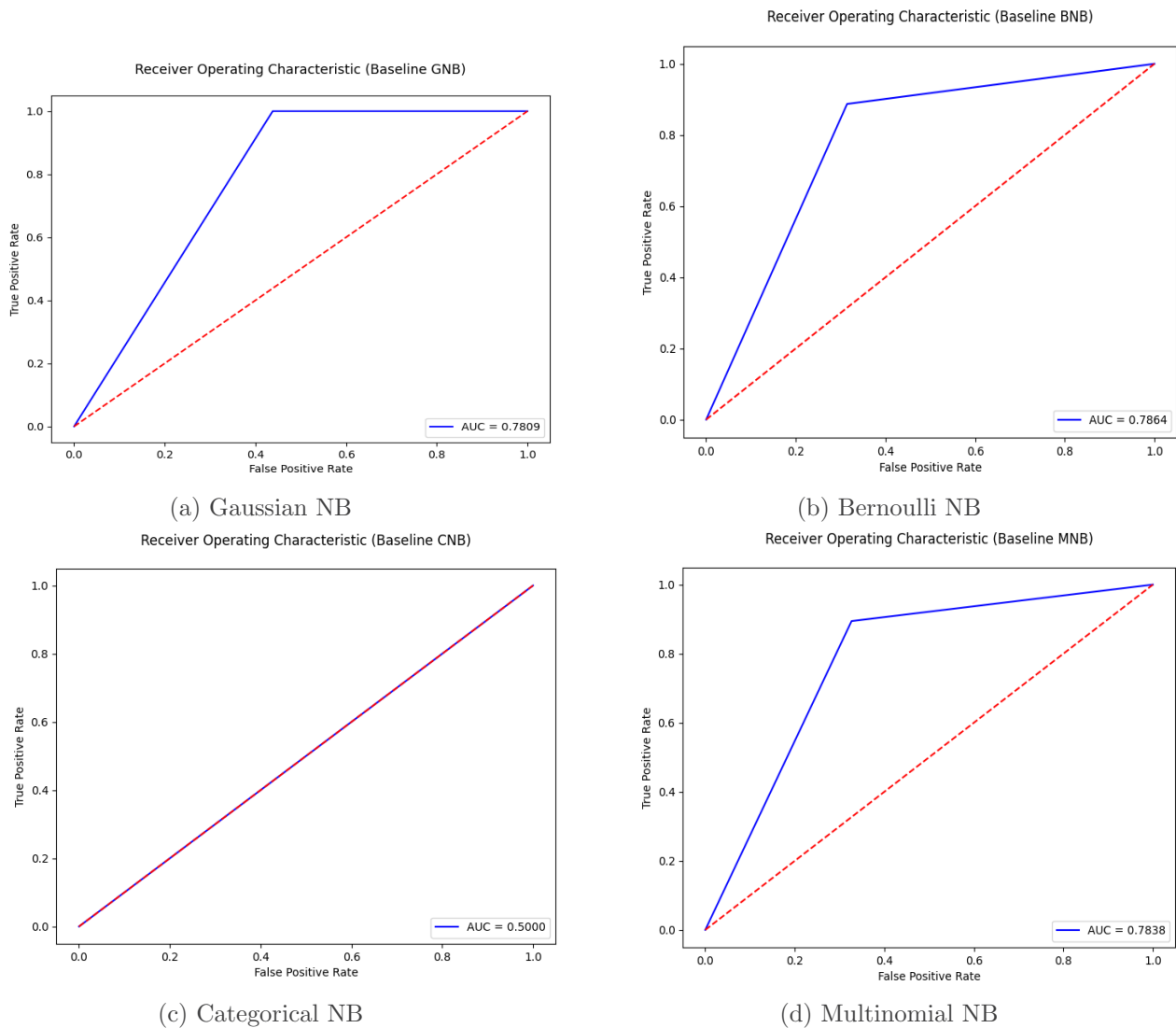


Figure 6.6: ROC curves for different NB classifiers using *TF-IDF*

As discussed in 3.9, AUC is a rank-based metric used for the comparison of different classifiers to evaluate their overall performance. Figures 6.7 and 6.8 show the ROC curves and AUC scores for the different NB classifiers implemented in this work. Multinomial NB is relatively the best-performing NB classifier, as indicated by the AUC score. Table 6.2 presents a comparison of the performance evaluation metrics of the implemented NB classifiers.

NB Classifiers	Accuracy (%)	Precision (%)	Recall (%)	F-score (%)	ERR (%)	TNR (%)	FPR (%)	AUC (%)
Gaussian NB	77.8	84.7	77.8	76.7	22.2	56.0	44.0	78.1
Bernoulli NB	79.3	80.9	79.3	79.1	20.7	69.0	31.0	79.5
Categorical NB	49.5	24.5	49.5	32.7	50.5	0.0	100	50
Multinomial NB	83.8	85.3	83.8	83.7	16.2	74.0	26.0	83.9

Table 6.2: Comparison of different NB classification performance evaluation metrics using *Tokenizer*

NB Classifiers	Accuracy (%)	Precision (%)	Recall (%)	F-score (%)	ERR (%)	TNR (%)	FPR (%)	AUC (%)
Gaussian NB	77.9	84.7	77.9	76.8	22.1	56.0	44.0	78.1
Bernoulli NB	78.1	79.8	78.5	78.3	21.9	69.0	31.0	78.6
Categorical NB	49.5	24.5	49.5	32.7	50.5	0.0	100	50
Multinomial NB	78.3	79.8	78.3	78.0	21.7	67.0	33.0	78.4

Table 6.3: Comparison of different NB classification performance evaluation metrics using *TF-IDF*

The MNB classifiers are selected as the baseline NB classifiers to be tuned, because of their better performance results. After the training of the tuned MNB models with hyperparameters derived from algorithm 8. The performance of the tuned MNB is evaluated with the testing dataset. Figures 6.7a and 6.7b show the confusion matrix and the ROC curve respectively for the tuned MNB classification performance evaluation.

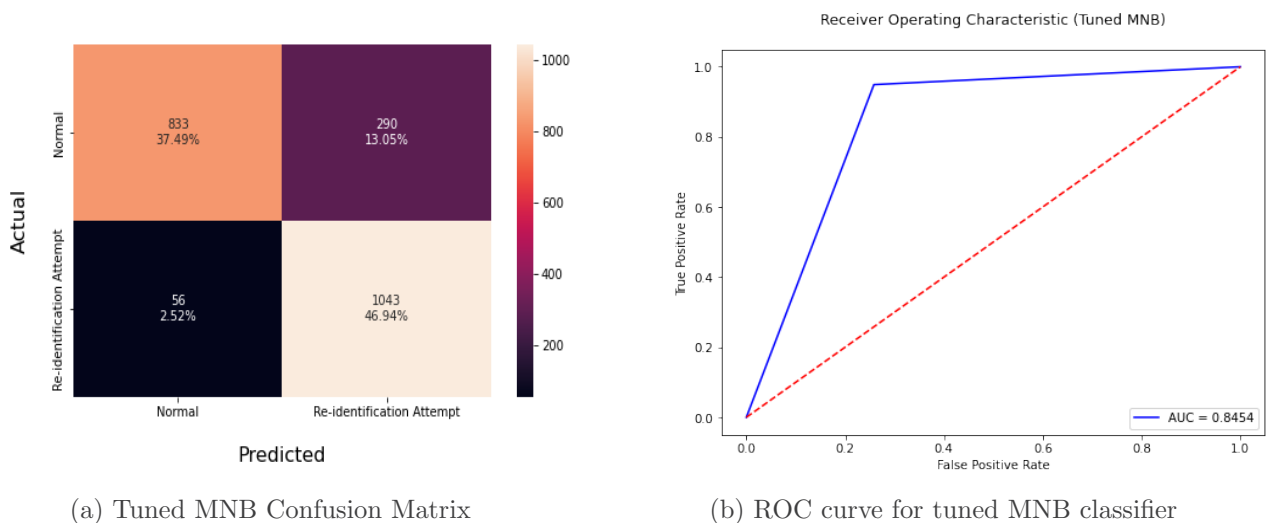


Figure 6.7: Tuned MNB classification performance evaluation confusion matrix and ROC curve (*Tokenizer*)

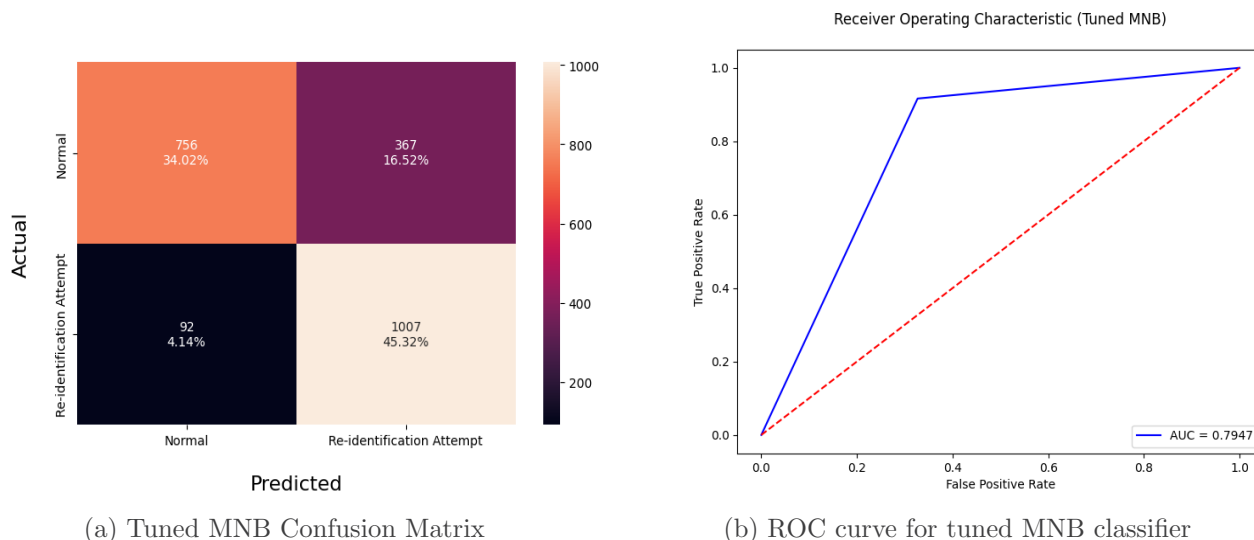


Figure 6.8: Tuned MNB classification performance evaluation confusion matrix and ROC curve (*TF-IDF*)

The tuned MNB classifier correctly classified 1,043 and 833 re-identification attempts and normal samples respectively. This is an improvement compared to the 1,031 correctly classified re-identification samples and 831 correctly classified normal samples from the baseline MNB. The misclassification in the tuned MNB is also reduced to 56 re-identification instances confused as normal and 290 normal instances confused as re-identification. This is an improvement from baseline MNB with 68 and 292 misclassified samples for re-identification and normal instances respectively. Table 6.4 presents a comparison of the classification performance of the baseline and the tuned MNB. Being the best performing Naive Bayes model for the classification of the research dataset, the tuned MNB is selected as the NB classifier to best evaluated against other algorithms (MLP, KNN, and LR) implemented in this work.

MNB Classifiers	Accuracy (%)	Precision (%)	Recall (%)	F-score (%)	ERR (%)	TNR (%)	FPR (%)	AUC (%)
Baseline MNB	83.8	85.3	83.8	83.7	16.2	74.0	26.0	83.9
Tuned MNB	84.4	86.1	84.4	84.3	15.6	74.0	26.0	84.5

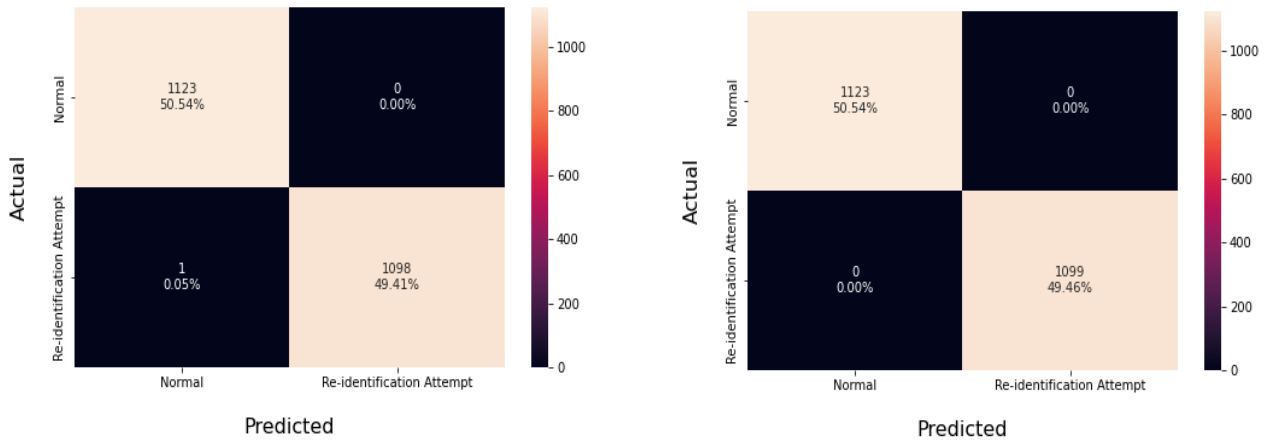
Table 6.4: Comparison of baseline and tuned MNB classification performance evaluation metrics (*Tokenizer*)

MNB Classifiers	Accuracy (%)	Precision (%)	Recall (%)	F-score (%)	ERR (%)	TNR (%)	FPR (%)	AUC (%)
Baseline MNB	78.3	79.8	78.3	78.0	21.7	67.0	33.0	78.4
Tuned MNB	79.3	81.3	79.3	79.1	20.7	67.0	33.0	79.5

Table 6.5: Comparison of baseline and tuned MNB classification performance evaluation metrics (*TF-IDF*)

6.1.3 Evaluation of Re-identification Query classification with K-Nearest Neighbors

The classification performance evaluation for the KNN classifiers trained in 5.3.3 and 5.4.3 are represented in the confusion matrices shown in Figures 6.9 and 6.10 below. Using the *Tokenizer* features, the KNN with Euclidean distance as shown in 6.9a correctly saves all the 1,123 normal query samples as normal and correctly saves 1,098 of 1,099 re-identification attempt samples as such, confusing one re-identification attempt instance as normal. With *TF-IDF*, Euclidean distance KNN correctly saved all but one sample in both the normal and re-identification classes, as depicted in figure 6.10a. Both the *Tokenizer* and *TF-IDF* Euclidean KNN models recorded a testing accuracy, precision, recall, f-score, and AUC score of 99.9%. The full classification reports for the KNN models with Euclidean distance are in Appendices D.1.9 and D.3.9. The KNN models with Manhattan distance (*Tokenizer* and *TF-IDF*) correctly save all query instances from each class to their corresponding classes, achieving a classification score of 100% for the accuracy, precision, recall, f-score, and AUC metrics.



(a) KNN with Euclidean distance

(b) KNN with Manhattan distance

Figure 6.9: Classification performance evaluation confusion matrices for KNN with different distance metric (*Tokenizer*)

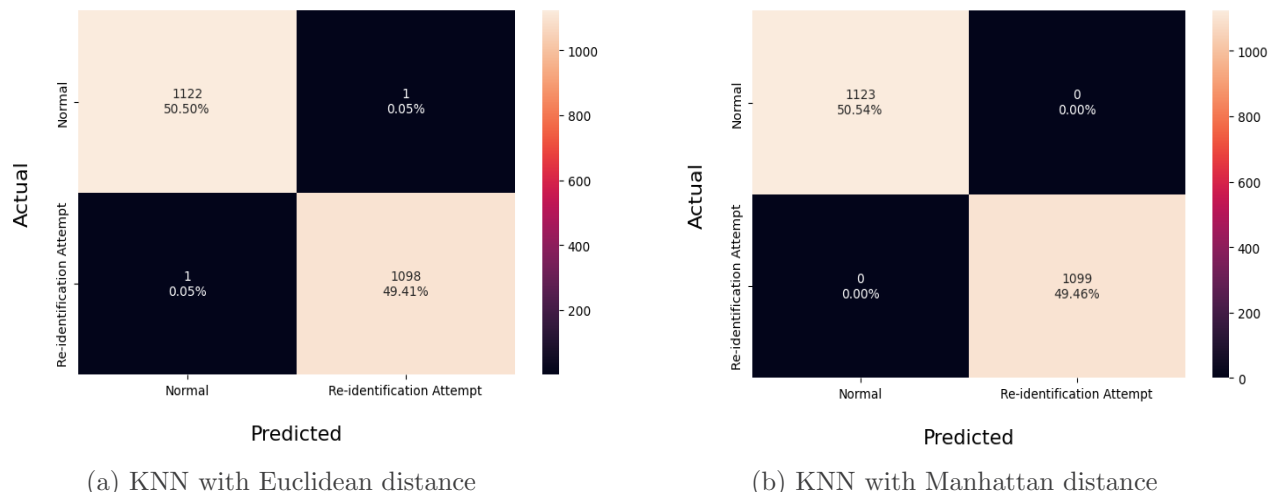


Figure 6.10: Classification performance evaluation confusion matrices for KNN with different distance metric (*TF-IDF*)

The ROC curves showing the AUC scores for Euclidean and Manhattan distance metric-trained KNN classifiers are shown in Figures 6.11 and 6.12.

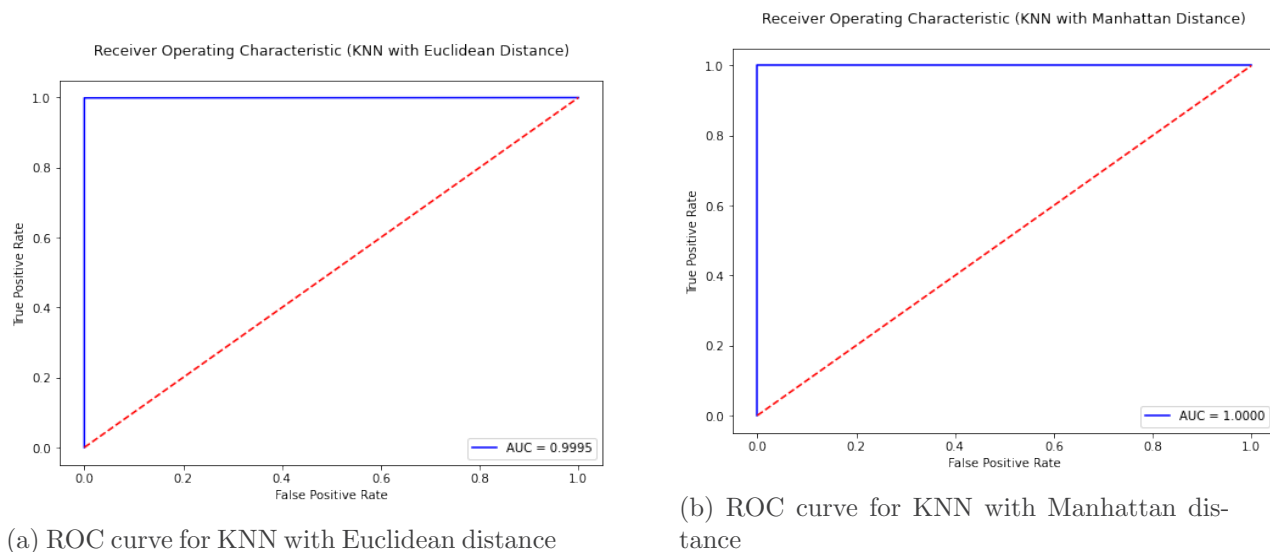
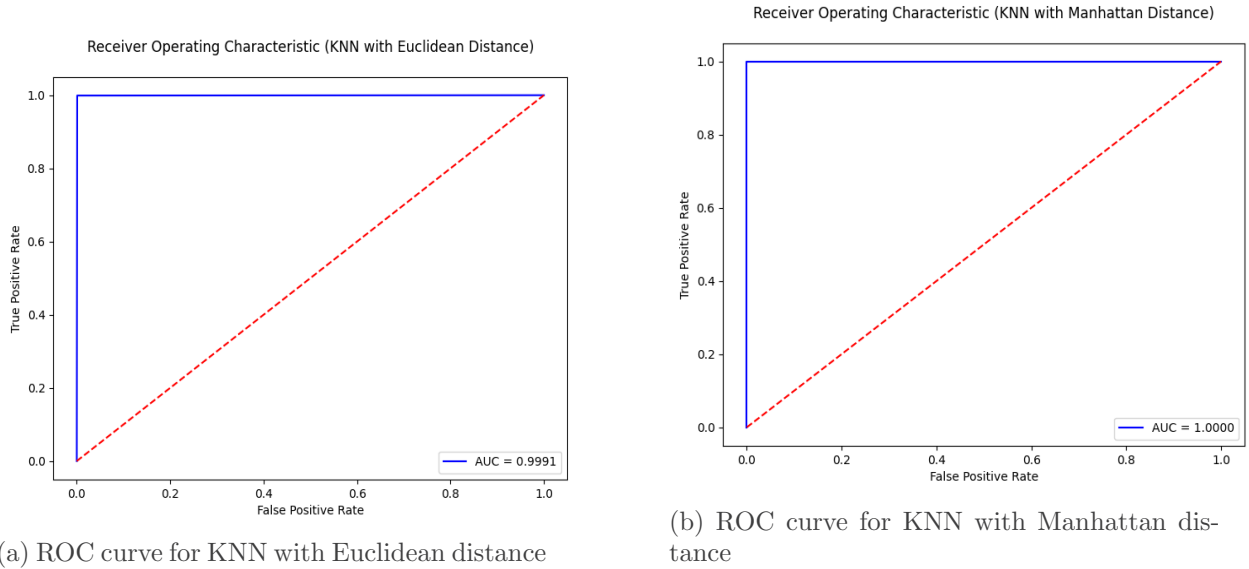


Figure 6.11: ROC curves for KNN with different distance metric (*Tokenizer*)

Table 6.6 below presents the comparison of two KNN models trained with Euclidean and Manhattan distance functions. The full classification reports for the two KNN classifiers are in appendices D.1.9 and D.1.10.

The evaluation metrics scores in the KNN classifiers suggest high performance with both the Euclidean and Manhattan distance functions. These performance evaluation scores are a result of KNN being an instance-based algorithm (lazy learning) that postpones computations until it is tested with a new, unseen instance. The high evaluation performance in KNN regardless of the distance function used or feature extraction implemented can be attributed to this characteristic

Figure 6.12: ROC curves for KNN with different distance metric (*TF-IDF*)

KNN Classifiers	Accuracy (%)	Precision (%)	Recall (%)	F-score (%)	ERR (%)	TNR (%)	FPR (%)	AUC (%)
Euclidean KNN	99.9	99.9	99.9	99.9	0.1	100	0	99.9
Manhattan KNN	100	100	100	100	0	100	0	100

Table 6.6: Comparison of classification performance evaluation metrics in KNN with Euclidean and Manhattan distance functions (*Tokenizer*)

KNN Classifiers	Accuracy (%)	Precision (%)	Recall (%)	F-score (%)	ERR (%)	TNR (%)	FPR (%)	AUC (%)
Euclidean KNN	99.9	99.9	99.9	99.9	0.1	100	0	99.9
Manhattan KNN	100	100	100	100	0	100	0	100

Table 6.7: Comparison of classification performance evaluation metrics in KNN with Euclidean and Manhattan distance functions (*TF-IDF*)

in the KNN algorithm.

6.1.4 Evaluation of Re-identification Query classification with Logistic Regression

As discussed in section 5.3.4, using *Tokenizer* features, a baseline logistic regression model is trained with default hyperparameters that use a C value of 1.0 and the *lbfgs* solver. The performance evaluation of baseline LR with the testing data showed that 964 re-identification attempt queries are correctly classified (43.38% of the entire testing set) and 48.15% of normal samples in the test set are correctly classified are normal queries, this comprises 1,070 query samples. The misclassification in the baseline LR model shows 136 re-identification query samples confused as

normal and 53 normal samples wrongly classified as re-identification attack attempts. The performance evaluation of the model achieved an accuracy, recall, f-score, AUC score of 91.5%, and a precision of 91.8%. With *TF-IDF* features, 1,037 (46.67%) re-identification samples are correctly classified, and 980 (44.1%) normal are correction classified, showing a higher true positive rate and a lower true negative rate than the *Tokenizer* trained LR model. With 143 normal query samples classified as re-identification and 62 re-identification samples classified as normal, the model shows lower false negative rates and higher false positive rates than when trained with *Tokenizer* features. The performance evaluation of the *TF-IDF* model achieved an accuracy, recall, f-score, AUC score of 90.8% and a precision of 90.9%. The confusion matrix representing the evaluation of the performance in both baseline logistic regression classifiers is shown in Figures 6.13a and 6.14a.

Tuned versions of the LR classifiers are then trained with modifications to the *C* and *solver* values, changing them to 0.0001 and *newton-cg* respectively. The performance evaluation of the tuned LR classifier resulted in all samples correctly classified, showing all 1,099 re-identification query samples predicted as such and the 1,123 normal samples in the testing set predicted as members of the normal class. This is the case in both *Tokenizer* and *TF-IDF* feature-trained models. The tuned LR models achieved accuracy, precision, recall, f-score, and AUC score of 100%. Figures 6.13b and 6.14b show the confusion matrices to represent the classification performance of the tuned LR models.

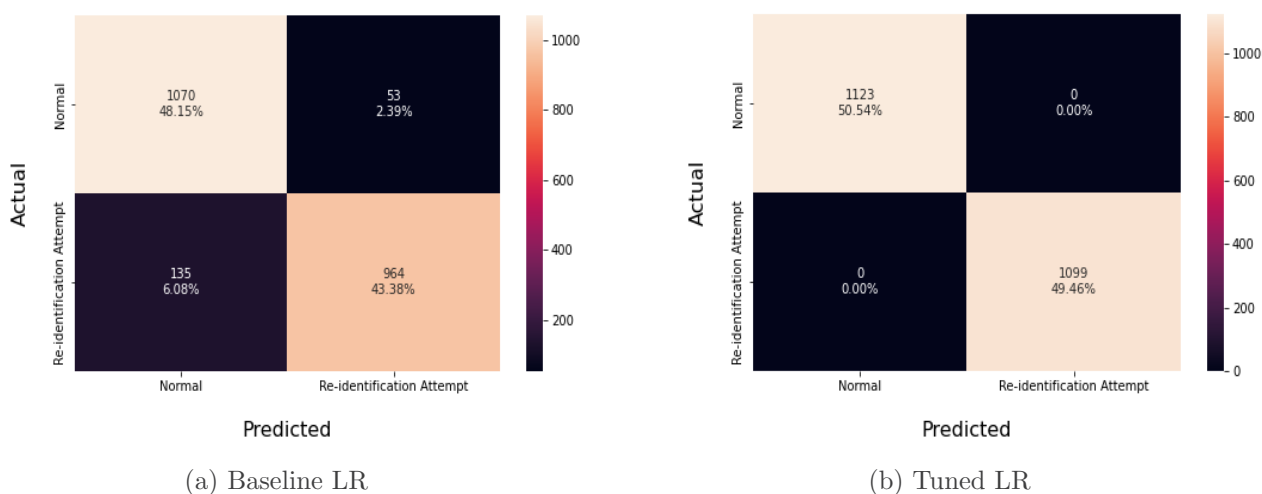


Figure 6.13: Classification performance evaluation confusion matrices for baseline and tuned Logistic Regression (*Tokenizer*)

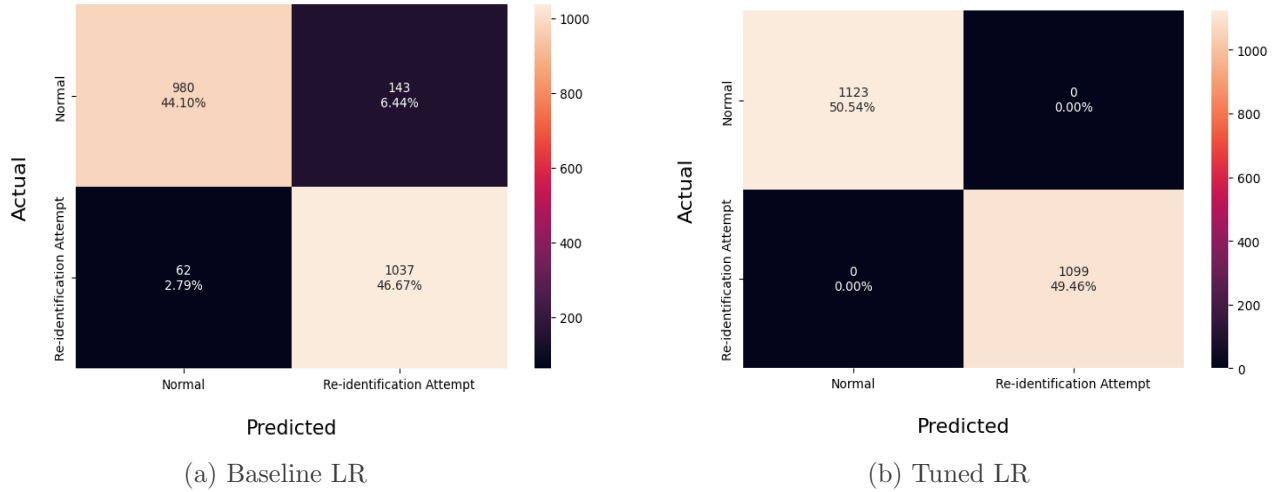


Figure 6.14: Classification performance evaluation confusion matrices for baseline and tuned Logistic Regression (*TF-IDF*)

For comparison of model performance on the testing dataset, ROC curves showing the AUC scores for baseline and tuned logistic regression models are depicted in Figures 6.15 and 6.16.

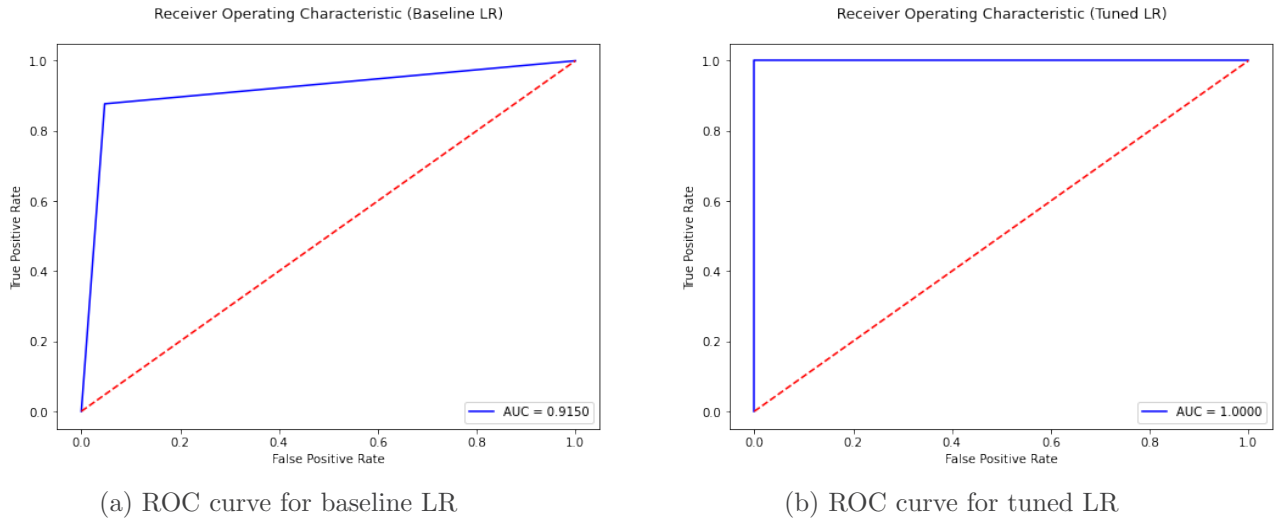


Figure 6.15: ROC curves for baseline and tuned LR classifiers (*Tokenizer*)

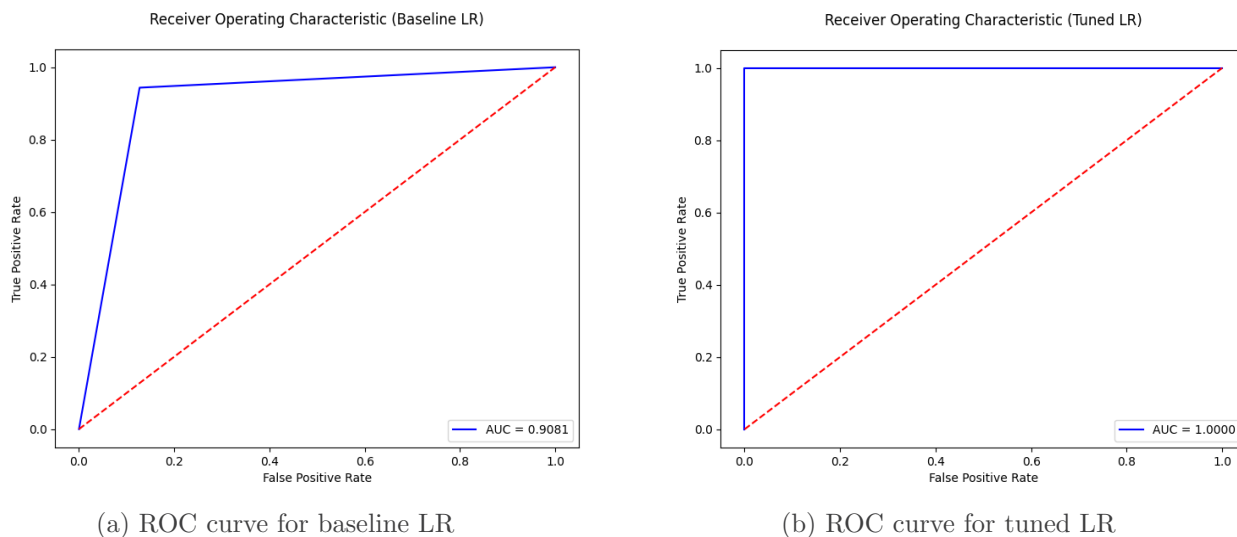


Figure 6.16: ROC curves for baseline and tuned LR classifiers (*TF-IDF*)

Tables 6.8 and 6.9 depict the classification performance evaluation comparison between the baseline and tuned LR models using both *Tokenizer* and *TF-IDF*.

LR Classifiers	Accuracy (%)	Precision (%)	Recall (%)	F-score (%)	ERR (%)	TNR (%)	FPR (%)	AUC (%)
Baseline LR	91.5	91.8	91.5	91.5	8.5	95	5	91.5
Tuned LR	100	100	100	100	0	100	0	100

Table 6.8: Comparison of baseline and tuned LR classification performance evaluation metrics (*Tokenizer*)

LR Classifiers	Accuracy (%)	Precision (%)	Recall (%)	F-score (%)	ERR (%)	TNR (%)	FPR (%)	AUC (%)
Baseline LR	90.8	90.9	90.8	90.8	9.2	87	13	90.8
Tuned LR	100	100	100	100	0	100	0	100

Table 6.9: Comparison of baseline and tuned LR classification performance evaluation metrics (*TF-IDF*)

6.2 Model Performance Evaluation Comparison

Sections 5.3 and 5.4 detailed the exploration of four different binary classification models for the prediction of SQL query patterns attempting re-identification attacks on anonymised databases, using features from two different (*Tokenizer* and *TF-IDF*) feature extraction methods. From the evaluation of the classification capabilities of the models presented in section 6.1, the KNN with Manhattan distance function, logistic regression with tuned hyperparameters, multilayer perceptron, and the tuned multinomial naive Bayes are the best-performing models. The comparison

of these four models is in Table 6.10. The comparison shows that KNN, LR, and MLP have a 0% false positive rate, suggesting that these classifiers do not confuse normal query instances as re-identification attack attempts. The MNB makes a false positive misclassification 26% of the time with *Tokenizer* and 33% with *TF-IDF*. The comparison also shows that the *TF-IDF* trained MLP has an improved performance over the *Tokenizer* trained MLP.

Classifiers	Accuracy (%)	Precision (%)	Recall (%)	F-score (%)	ERR (%)	TNR (%)	FPR (%)	AUC (%)
KNN	100	100	100	100	0	100	0	100
LR	100	100	100	100	0	100	0	100
MLP	98.7	98.7	98.7	98.7	1.3	100	0	98.7
MNB	84.4	86.1	84.4	84.3	15.6	74.0	26.0	84.5

Table 6.10: Comparison of performance evaluation metrics in re-identification attack attempt recognition models (*Tokenizer*)

Classifiers	Accuracy (%)	Precision (%)	Recall (%)	F-score (%)	ERR (%)	TNR (%)	FPR (%)	AUC (%)
KNN	100	100	100	100	0	100	0	100
LR	100	100	100	100	0	100	0	100
MLP	100	100	100	100	0	100	0	100
MNB	79.3	81.3	79.3	79.1	20.7	67.0	33.0	79.5

Table 6.11: Comparison of performance evaluation metrics in re-identification attack attempt recognition models (*TF-IDF*)

6.3 Understanding Misclassification in the Models

Various factors can contribute to the occurrence of misclassification in machine learning classification models. These factors include inaccurate or insufficient training datasets, the implemented feature extraction technique, the choice of classifier algorithms, and hyperparameters tuning. Further exploring the dataset in this research, this section provides insights into the occurrence of misclassification in the models as they learn to recognise and classify SQL queries attempting re-identification attacks. This section provides insights into understanding the possible reasons behind the misclassifications that occurred in the experiment presented in section 6.1.

Figure 6.17 below presents a word cloud depiction of the words in the research dataset after augmentation. The words coloured in red are the most frequent words belonging to both the normal and the re-identification attack attempt classes. The yellow-coloured words are the most frequent in the attack class and the green words are those that are most frequent in the normal class. The words coloured white are infrequent words with limited occurrence. There is a high probability that these words with limited occurrence are regarded as insignificant by the models during classification. For example, the token *COUNT* is an SQL query keyword that is coloured white in the depicted word cloud, suggesting that its frequency of occurrence is limited. From the knowledge and context of the research dataset, there is a high likelihood that a user will use the

There are 738 words that belong to both classes, 1,127 words belong to only the normal class, and 49 words belong to only the re-identification attack class. There is a significant number of words present in both classes and this will negatively affect the ability of the models to accurately classify samples.

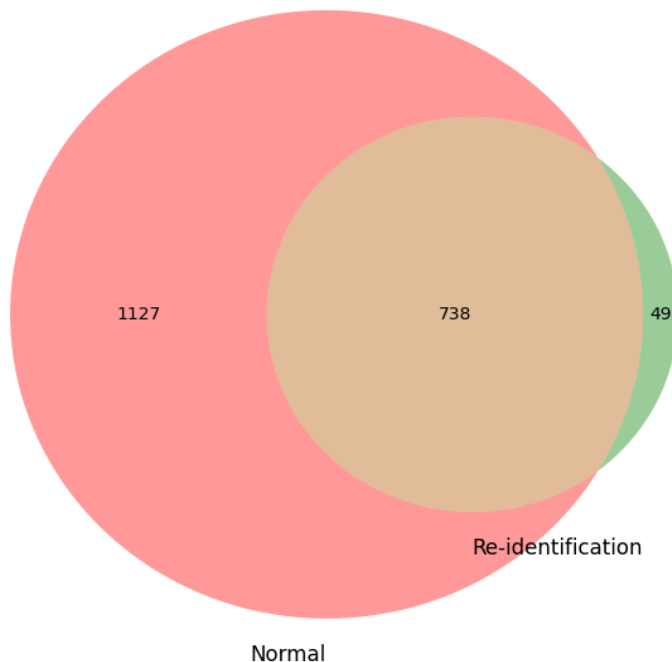


Figure 6.18: Venn diagram showing the number of words in each class

6.4 Unseen Data Test

The discussion in section 3.9.1 emphasised that the true measure of a classifier is the ability to *generalise*, using the learned patterns and principles from the training dataset to recognise and correctly predict class membership for a new, previously unseen data sample. A classifier that is capable of generalising and correctly predicting class membership for unseen data samples is practicable and can be applied in the real world to solve classification problems. This section presents the testing of the classifiers trained in this research using new instances of re-identification attack attempt queries and normal queries. It is important to evaluate the classifiers' ability to generalise and classify unseen query samples. The SQL data collection campaign for this research as detailed in section 4.6, used the anonymised synthetic database created for the Netflix Prize Data re-identification scenario described in section 4.3.2. The SQL queries (normal and re-identification attempts) used in the training of classification algorithms are gathered using the NPD database. New unseen test samples are collected using the same NPD database to test for generalisation ability in the classifiers. However, to further test how well the classifiers developed in this research work can generalise, unseen test samples are also collected using the anonymised synthetic database created for the GIC re-identification scenario in section 4.3.2, for the GIC dataset. This is to test and evaluate the ability of the classifiers to recognise re-identification query patterns from multiple domains. A total of 28 new unseen query samples are collected for model testing, 14 from each anonymised database (NPD and GIC database). 7 samples represent

users executing normal queries and another seven are re-identification attack attempt instances. The trained and saved MLP, MNB, KNN, and LR models are tested with the unseen SQL queries using the steps depicted in Algorithm 11 below.

Algorithm 11 Model Testing Algorithm

1. Create a regular expression pattern using Python raw string notation
2. Iterate through the rows, columns, and cells of *.csv* files. Such that if cell data is valid (not null), it is converted to a string and then stored to a *list_of_words* variable
3. Use regular expression pattern to add space around special characters
4. Split spaced-out row data into words and add to *list_of_words*
5. Iterate through *list_of_words* and add words to a new *clean_words* variable. Ignore null spaces, removing “ ’ ” and replacing numbers with $\langle NUM \rangle$
6. Return *clean_words*
7. Load the *tokenizer* created in Algorithm 5
8. Read an unseen test *.csv* query file and append the words in it to *X_test* variable
9. Use the *tokenizer* to convert the words *X_test* into a matrix of numerical features using frequency count and save to *X_predict*
10. Load a saved classifier model
11. Predict the class membership of a new sample with *X_predict*.
12. Set a binary classification threshold, such that the test sample is classified as *attacking* (re-identification attempt) if the prediction score is ≥ 0.5 else *normal*

To test the models developed using *TF-IDF*, step 7 of algorithm 11 loads the *tfidf* vectorizer created in algorithm 6. At step 9, testing with *TF-IDF* models requires the testing algorithm to create an *X_predict* data frame and convert the data matrix into a dense array representation.

The model testing algorithm depicts the process of predicting if a new sample file containing a series of SQL queries is attempting a re-identification attack or not. The process begins with cleaning and preparing the new test sample to be passed to the models for prediction. The words in the sample are tokenized and encoded to numerical features, and the *tokenizer* and *tfidf* vectorizer created during feature extraction in Algorithms 5 and 6 are used in the prediction process to compare the features in a new sample to the features learned by the models from the training dataset. Based on frequency count, and feature importance (in the case of *TF-IDF*), the model predicts the class label that has more features in common with the new test sample. As shown in step 3 of Algorithm 2, the normal query class has the label 0, and label 1 is assigned the re-identification attempt query class. The threshold for testing is set at 0.5, such that a prediction estimation below 0.5 is classed as a normal query and estimations greater than or equal to 0.5 are classified as a re-identification attempt. The closer the prediction estimation score is to the class label, the higher the *degree of confidence* the classifier has in the prediction for any particular test sample.

The same strategy employed in section 6.1 for model performance evaluation with the test dataset is adopted for the unseen data test. This involves using the unseen test data to evaluate and compare the performance of baseline and tuned versions of the Multinomial NB and Logistic Regression classifier, the KNN classifiers with Euclidean and Manhattan distance, and the Multilayer Perceptron models. This is to determine which of the classifiers has learned the principles of the training dataset the best and can generalise on unseen test samples. Also, this determines if there are correlations between the performance of the models on the test dataset and the unseen test samples.

The data collection method adopted to gather unseen query samples for testing in this work involves having a participant explore both databases (NPD and GIC) to generate normal SQL query samples, and then the same individual proceeds to attempt a re-identification attack on the databases. Therefore, normal and re-identification attempt query sample 1 for NPD and for GIC are from the same participants. This method is the same for all other test samples. As a result, the unseen data test will not only look for correlations in the test results based on consistency among different classifiers but also among different samples.

6.4.1 Multilayer Perceptron

Tokenizer

Table 6.12 shows that the MLP classifier trained with *ToTokenizer* features is able to recognise and predict 10 out of 14 re-identification attempt queries as attacking queries. The classifier recognised attack queries from different database domains, with three recognised samples from the NPD database and the other three from the GIC database. Unseen re-identification attack attempt query samples 1, 4, 5, 6, and 7 from the NPD database have prediction estimations of 0.743, 0.999, 0.994, 0.794, and 0.997 respectively. Attack samples 1, 4, 5, 6, and 7 executed against the GIC database have a prediction value of 0.969, 0.989, 0.967, 0.989, and 0.999 respectively. These prediction scores are over the 0.5 classification threshold, thereby recognised and classified as re-identification attack queries. Four samples, samples 2 and 3 from both the NPD and GIC databases are predicted as normal, even though participants in the unseen test data collection were attempting re-identification. All the normal unseen test samples are all recognised and classified as non re-identification attack queries, with prediction estimations lower than the threshold.

Attack Samples	MLP	
	NPD	GIC
Sample 1	0.743	0.969
Sample 2	0.450	0.234
Sample 3	0.129	0.174
Sample 4	0.999	0.989
Sample 5	0.994	0.967
Sample 6	0.794	0.989
Sample 7	0.997	0.999
Normal Samples		
Sample 1	0.001	0.045
Sample 2	0.003	0.083
Sample 3	0.000	0.049
Sample 4	0.000	0.089
Sample 5	0.0004	0.083
Sample 6	0.0005	0.146
Sample 7	0.002	0.076

Table 6.12: *Tokenizer* MLP classifier unseen data test result

The unseen data test results in the MLP show consistency with the classifier’s performance on the test set during the performance evaluation presented in section 6.1.1. The performance evaluation showed a TNR of 100%, suggesting that all normal samples are correctly classified. The unseen data test shows the same results, with all unseen normal samples predicted as normal queries. There is a 1.3% error rate in the classifier, suggesting that it may occasionally misclassify query samples, however, at a low rate. The performance evaluation and unseen data test in the MLP classifier suggest that the model has good and practicable generalisation ability, recognising new instances of re-identification attack attempts approximately 71% of the time.

TF-IDF

As depicted in table 6.13, the MLP classifier trained with *TF-IDF* features shows an improved unseen data test performance. The classifier predicts 11 of 14 attack attempt samples as attacking queries, with a higher degree of confidence in each of the samples. From the NPD database, unseen re-identification attack attempt query samples 1, 2, 4, 5, and 7 are correctly predicted to be attack attempts. Attack samples 1, 2, 4, 5, 6, and 7 from the GIC database are also predicted correctly. Attacking sample 2 of the NPD database was misclassified as normal in the *Tokenizer* MLP, however, it was correctly classified with *TF-IDF*. This is also the case for attack sample 2 from the GIC database. It is also noted that attack sample 6 is misclassified with *TF-IDF* and correctly predicted with *Tokenizer*. All normal samples are correctly classified.

Attack Samples	MLP	
	NPD	GIC
Sample 1	0.999	0.999
Sample 2	0.756	0.998
Sample 3	0.000	0.005
Sample 4	1.0	0.999
Sample 5	1.0	0.999
Sample 6	0.000	0.999
Sample 7	0.994	1.0
Normal Samples		
Sample 1	0.000	0.000
Sample 2	0.000	0.0001
Sample 3	0.000	0.000
Sample 4	0.000	0.0007
Sample 5	0.000	0.0004
Sample 6	0.000	0.0006
Sample 7	0.000	0.0001

Table 6.13: *TF-IDF* MLP classifier unseen data test result

The results in the unseen data test for *TF-IDF* MLP show a high generalisation capability with an accurate prediction of new query samples approximately 79% (11 of 14) of the time.

6.4.2 Multinomial Naive Bayes

Tokenizer

As presented in Table 6.14, the baseline MNB classifier predicts all the 14 re-identification attack attempt samples as attacking queries, with prediction estimation values equal to or over the 0.5 threshold. The classifier recognises re-identification attack attempts across the two different databases. However, there is a high false positive rate in the classifier. Ten of the normal query samples are predicted as an attack attempt instance, this is the case for 6 out of 7 normal samples from the NPD database and 4 normal samples from the GIC database. As discussed in the performance evaluation of the MNB classifier in section 6.1.2, the tuned MNB classifier shows a slight improvement in performance. This is also the case with unseen data testing. The tuned MNB predicted 13 of the 14 re-identification test samples as a re-identification attack attempt with a slightly higher degree of confidence in comparison to the baseline model. Attack Sample 3 from the GIC database is estimated at a 0.488 prediction score by the tuned MNB, suggesting it is a normal instance. The FPR in the tuned MNB is lower than in the baseline version. Similar to the baseline model, tuned MNB predicts most normal samples from the NPD database as attacking, and 5 of the 7 normal samples from the GIC database are predicted as normal. Overall, the unseen test performance in the baseline and tuned MNB shows that both models have similar re-identification attack attempt recognition capabilities. However, the tuned MNB shows a better generalisation ability between the two models. This can be attributed to the tuned model showing a higher degree of confidence in the recognised attack attempt queries and a lower FPR.

Attack Samples	Baseline MNB		Tuned MNB	
	NPD	GIC	NPD	GIC
Sample 1	0.522	0.547	0.54	0.534
Sample 2	0.539	0.513	0.569	0.503
Sample 3	0.532	0.5	0.544	0.488
Sample 4	0.619	0.571	0.728	0.567
Sample 5	0.584	0.549	0.666	0.537
Sample 6	0.516	0.533	0.527	0.539
Sample 7	0.541	0.538	0.575	0.568
Normal Samples				
Sample 1	0.526	0.498	0.517	0.486
Sample 2	0.519	0.516	0.513	0.507
Sample 3	0.521	0.493	0.507	0.482
Sample 4	0.527	0.506	0.519	0.495
Sample 5	0.509	0.497	0.502	0.488
Sample 6	0.476	0.522	0.457	0.512
Sample 7	0.539	0.501	0.538	0.488

Table 6.14: Unseen data test result for baseline and tuned MNB classifiers (*Tokenizer*)

The performance evaluation of the tuned MNB shows a 15.6% error rate, a 26% FPR, and an accuracy of 84.4%, this suggests that there is a relatively significant misclassification expectation, however, the subpar performance shown by the MNB classifier on unseen data points to high variance (overfitting) in the classifier. The MNB performance on the unseen test samples can be attributed to the simplicity of the Naive Bayes model algorithm. The naive assumptions made by Naive Bayes models contribute to their inaccuracies in many real-world applications. This proves to be the case in this research for the application of Multinomial NB in the recognition of re-identification attack patterns in SQL queries.

TF-IDF

Similar to the unseen data test performance recorded in table 6.15 for the *Tokenizer* feature trained MNB, the *TF-IDF* trained baseline MNB classified 13 out of the 14 attack samples as instances of re-identification attack attempts (with a higher degree of confidence). Attack sample 2 from the GIC database is predicted to be normal in this classifier. 11 out of the 14 normal query samples returned prediction scores over 0.5, suggesting that they are attack samples. This suggests a high FPR in the classifier. A more consistent and improved performance is recorded for the tuned version of this classifier. The test results show that 10 of 14 attack attempt samples are correctly classified, a 71% unseen data test accuracy. Attack samples 2 and 3 from both the NPD and GIC databases are misclassified as normal. The tuned MNB model recognised all 14 normal query samples as normal, suggesting low FPR.

Attack Samples	Baseline MNB		Tuned MNB	
	NPD	GIC	NPD	GIC
Sample 1	0.982	0.872	0.959	0.984
Sample 2	0.932	0.297	0.379	0.005
Sample 3	0.938	0.629	0.458	0.180
Sample 4	0.999	0.911	0.999	0.991
Sample 5	0.998	0.879	0.999	0.985
Sample 6	0.831	0.969	0.556	0.966
Sample 7	0.977	0.986	0.985	0.998
Normal Samples				
Sample 1	0.468	0.601	0.009	0.157
Sample 2	0.556	0.678	0.006	0.017
Sample 3	0.648	0.591	0.274	0.156
Sample 4	0.719	0.671	0.430	0.202
Sample 5	0.573	0.759	0.051	0.188
Sample 6	0.097	0.660	0.001	0.016
Sample 7	0.315	0.640	0.001	0.189

Table 6.15: Unseen data test result for baseline and tuned MNB classifiers (*TF-IDF*)

21.7% and 33% error rate and FPR respectively are recorded for the *TF-IDF* tuned MNB model, these are higher than those recorded in the *Tokenizer* tuned MNB. However, the unseen data test performance result for the *TF-IDF* feature trained tuned Multinomial Naive Bayes implies that it is the best performing Naive Bayes classifier in this work. The result shows real-world applicability and no sign of overfitting as recorded in the other MNB models.

6.4.3 K-Nearest Neighbors

Tokenizer

The unseen data testing in Table 6.16 presents the test result for KNN with Euclidean and Manhattan distance functions. The performance evaluation presented for these classifiers in section 6.1.3 shows that accuracy scores of 99.9% and 100% were achieved by the KNN with Euclidean and the KNN with Manhattan distance functions respectively. Given that a KNN classifier is a lazy learning algorithm (simply memorising the training dataset), all computation is performed by the classifier at this stage of new unseen data testing. The Euclidean distance-trained KNN recognises re-identification attack samples 1, 2, 3, and 6 from the NPD database as normal while samples 4, 5, and 7 are predicted as re-identification attempts. 6 out of 7 attacking samples from the GIC database are recognised as attack attempts. All but one (Sample 4 from the GIC database) normal query samples are predicted to be normal database queries, showing a high TNR and a low FPR in the classifier. This result suggests that the classifier is capable of generalising unseen samples. The KNN trained with the Manhattan distance function on the other hand shows a high testing error with the unseen dataset, which suggests an element of overfitting in the model, as the performance evaluation of this model shows 100% metrics scores. The model predicts 10 of 14 re-identification attack samples as attack instances, however, 7 of 14 normal instances are predicted as re-identification attempts. None of the normal queries from the GIC database is correctly predicted. The better performance results recorded in the KNN

with Euclidean distance function on unseen data is consistent with the claim in [164] that most lazy learner algorithms use Euclidean distance.

Attack Samples	KNN (Euclidean)		KNN (Manhattan)	
	NPD	GIC	NPD	GIC
Sample 1	0.4	1.0	0.0	1.0
Sample 2	0.4	0.0	1.0	0.0
Sample 3	0.2	0.6	0.8	1.0
Sample 4	1.0	1.0	1.0	1.0
Sample 5	1.0	1.0	1.0	1.0
Sample 6	0.4	0.8	1.0	1.0
Sample 7	0.8	0.8	0.2	1.0
Normal Samples				
Sample 1	0.0	0.4	0.0	1.0
Sample 2	0.2	0.2	0.0	0.8
Sample 3	0.0	0.4	0.0	1.0
Sample 4	0.0	0.6	0.0	1.0
Sample 5	0.0	0.4	0.0	0.8
Sample 6	0.2	0.2	0.0	1.0
Sample 7	0.2	0.4	0.2	1.0

Table 6.16: Unseen data test result for KNN classifiers with Euclidean and Manhattan distance functions (*Tokenizer*)

TF-IDF

As recorded in the KNN models trained with *Tokenizer* features, the performance evaluation for the Euclidean and Manhattan KNN models with *TF-IDF* shows 99% and 100% accuracy scores respectively. However, there are differences in the results obtained for the *TF-IDF* KNN models when tested with unseen data samples. The Euclidean KNN correctly recognised attack samples 4, 5, and 6 from the NPD database, and samples 3, 4, 6, and 7 from the GIC database are classified as re-identification attack attempts. This is a lower accurate prediction rate (7 out of 14) than the Euclidean KNN with *Tokenizer* features (9 of 14). When tested with the normal samples, all samples from the NPD database are correctly classified as normal. However, all the normal samples from the GIC database are misclassified to be attacking instances, suggesting a high FPR. This is the main difference in performance between the *Tokenizer* and *TF-IDF* KNN models (the classification of the normal samples from the GIC model using the Euclidean distance metric). The level of error recorded for the *TF-IDF* Euclidean distance KNN model points to overfitting in the model. The performance in the KNN model with Manhattan distance is consistent using both feature extraction methods, with high testing errors that suggest overfitting in the model.

Attack Samples	KNN (Euclidean)		KNN (Manhattan)	
	NPD	GIC	NPD	GIC
Sample 1	0.2	0.4	0.0	1.0
Sample 2	0.0	0.0	1.0	0.0
Sample 3	0.0	0.6	0.8	1.0
Sample 4	1.0	0.6	1.0	1.0
Sample 5	1.0	0.4	1.0	1.0
Sample 6	0.8	1.0	1.0	1.0
Sample 7	1.0	1.0	0.2	1.0
Normal Samples				
Sample 1	0.2	0.6	0.0	1.0
Sample 2	0.0	1.0	0.0	0.0
Sample 3	0.0	0.8	0.0	1.0
Sample 4	0.2	0.8	0.0	1.0
Sample 5	0.0	0.6	0.0	1.0
Sample 6	0.0	1.0	0.0	0.2
Sample 7	0.4	0.6	0.0	1.0

Table 6.17: Unseen data test result for KNN classifiers with Euclidean and Manhattan distance functions (*TF-IDF*)

6.4.4 Logistic Regression

Tokenizer

The prediction results from using the baseline and tuned LR classifiers to classify unseen query samples is presented in Table 6.18. The initial baseline classifier recognises 8 of 14 re-identification attempt queries as such, 3 from the NPD database and 5 from the GIC database. The normal query sample prediction test includes false positive results, with two normal samples (samples 3 and 4) from the NPD database predicted as re-identification attempts. The tuned LR unseen test result shows improvement in the performance of LR for re-identification attack query recognition. 11 out of 14 re-identification queries are recognised as re-identification attempts. This includes all 8 samples recognised by the baseline model (samples 4, 5, 6, and 7 from the NPD database and samples 1, 4, 5, 6, and 7 from the GIC database) and samples 1 and 3 from the NPD database, initially misclassified as normal by the baseline LR. The tuned LR also recognised the attack attempts with a higher degree of confidence in comparison to the baseline model. The tuned LR recorded no false positives, as all normal query samples are predicted as normal. The results in the tuned LR show that the model recognises re-identification query patterns in new instances approximately 79% of the time.

Attack Samples	Baseline LR		Tuned LR	
	NPD	GIC	NPD	GIC
Sample 1	0.389	0.66	0.999	1.0
Sample 2	0.443	0.409	0.026	0.0003
Sample 3	0.440	0.351	0.695	0.000
Sample 4	0.758	0.769	1.0	1.0
Sample 5	0.655	0.664	0.999	1.0
Sample 6	0.489	0.504	1.0	1.0
Sample 7	0.607	0.619	1.0	1.0
Normal Samples				
Sample 1	0.485	0.342	0.000	0.000
Sample 2	0.405	0.328	0.000	0.000
Sample 3	0.519	0.332	0.000	0.000
Sample 4	0.529	0.392	0.000	0.000
Sample 5	0.416	0.292	0.000	0.000
Sample 6	0.253	0.369	0.000	0.000
Sample 7	0.493	0.364	0.000	0.000

Table 6.18: Unseen data test result for baseline and tuned LR classifiers (*Tokenizer*)

The performance in the LR models against the unseen samples bears correlations to the performance results during evaluation with the testing set shown in section 6.1.4, with the tuned LR showing better performance metrics scores when compared to the baseline version.

TF-IDF

As presented in table 6.19, the baseline LR model with *TF-IDF* recognises 6 of 14 re-identification attempt queries correctly (4 from the NPD database and 2 from the GIC database). This is two correctly classified samples less than the baseline LR with *Tokenizer* feature extraction. However, there is consistency shown in the two baseline classifiers as the same 6 samples are classified as attack attempts. All the normal sample instances are correctly recognised as such. The *TF-IDF* tuned LR unseen test result records improvement in the performance of LR for re-identification attack query classification. 10 out of 14 re-identification queries are recognised as re-identification attempts. This includes all 6 samples recognised by the baseline model (samples 1, 4, 5, and 7 from the NPD database and samples 6 and 7 from the GIC database) and samples 1, 2, 4, and 5 from the GIC database, initially misclassified as normal by the baseline LR. The tuned LR also recognised the attack attempts with a higher degree of confidence in comparison to the baseline model. The tuned LR recorded no false positives, as all normal query samples are predicted as normal. The results in the tuned LR with *TF-IDF* show that the model recognises re-identification query patterns in new instances approximately 71% of the time.

Attack Samples	Baseline LR		Tuned LR	
	NPD	GIC	NPD	GIC
Sample 1	0.525	0.125	1.0	1.0
Sample 2	0.269	0.147	0.000	1.0
Sample 3	0.096	0.077	0.000	0.000
Sample 4	0.993	0.123	1.0	1.0
Sample 5	0.839	0.135	1.0	1.0
Sample 6	0.077	0.819	0.000	1.0
Sample 7	0.658	0.831	1.0	1.0
Normal Samples				
Sample 1	0.036	0.041	0.000	0.000
Sample 2	0.015	0.110	0.000	0.000
Sample 3	0.003	0.035	0.000	0.000
Sample 4	0.005	0.080	0.000	0.000
Sample 5	0.006	0.055	0.000	0.000
Sample 6	0.001	0.133	0.000	0.000
Sample 7	0.033	0.045	0.000	0.000

Table 6.19: Unseen data test result for baseline and tuned LR classifiers (*TF-IDF*)

6.5 Test Result Analysis

Using *Tokenizer* features, the MLP, tuned MNB, KNN with Euclidean distance metric, and tuned LR are the models with better generalisation attributes from the unseen data testing results. The results from using these four models to predict the class membership of unseen query samples is presented in Table 6.20. The MLP and LR classifiers show correlations in their results. First by predicting all normal query samples as normal and both classifiers showing 0% FPR on unseen data. This is an important characteristic in a classifier if it is to be implemented in a real-world environment. Also, both classifiers recognised re-identification attempts in the same 10 samples from the two databases (with high degrees of confidence). These are samples 1, 4, 5, 6, and 7 from both the NPD and the GIC database. The LR also predicted attack sample 3 from NPD as an attack instance. This consistency in the predictions on both the MLP and LR classifiers shows real-world practicability for these two models.

With *TF-IDF* as depicted in table 6.21 below, MLP, tuned MNB, KNN with Euclidean distance metric, and the tuned LR models remained the best-performing classifiers with unseen data testing. The MLP with *TF-IDF* showed improvement over the *Tokenizer* MLP. It recognises attack sample 2 from both the NPD and GIC databases, samples that the *Tokenizer* MLP misclassified as normal. However, the MLP with *TF-IDF* misclassified sample 6 from the NPD database, which is not the case in the *Tokenizer* MLP. Overall, both MLP models show significant consistency in the samples correctly classified, with the *TF-IDF* MLP showing a higher degree of confidence and slightly better performance. Even though the *TF-IDF* tuned LR classifier recognises re-identification attack samples with a higher degree of confidence than the *Tokenizer* LR, the overall performance with unseen data is slightly lower. However, there is consistency between the two classifiers as both recognises the same attack samples with discrepancies in samples 3 and 6 from the NPD database, correctly recognised by the tuned *Tonkenizer* LR but misclassified in the tuned LR with *TF-IDF*. The MLP and tuned LR with *TF-IDF* demonstrate

high real-world applicability with their unseen data test results.

The MLP and LR models' ability to consistently recognise re-identification patterns in SQL queries from a database different from the NPD database (used to generate the SQL training data) reinforce the proof of concept established in the methodology of this research, postulating that re-identification attacks on anonymised databases have a recognisable pattern and can be represented as a series of SQL queries.

As previously mentioned in this section, analysis based on test data samples can also be explored. Re-identification attack attempt in Sample 2 for both NPD and GIC databases are predicted as normal by the *Tokenizer* MLP and both LR classifiers, this suggests there is a possibility that the individual attempting the attack in these samples has limited technical skill to successfully execute a series of SQL query that points towards re-identification. Inference can also be made that there are prominent re-identification patterns in the queries in attack samples 1, 4, 5, 6, and 7 for both the NPD and GIC databases. The MLP and LR classifiers recognised these samples as re-identification with high degrees of confidence. This can be a result of the attacker's technical skill level to execute queries that are consistent with the re-identification of an anonymised database.

This research is confident that the MLP and the LR model are state-of-the-art classification models that can be implemented in real-world scenarios to recognise and predict if a series of SQL queries on an anonymised database is attempting a re-identification attack or not. The interpretation of the prediction estimation scores can also be tailored to fit a particular implementation domain. As reported by the work in [78], the implementation of the designed model involved using different thresholds on a 1 to 10 scale to decide the preferred action based on the anomaly score from any new query instance. The same method can be adopted for the implementation of the MLP and LR classifiers developed by this research. The unseen data test sets a threshold of 0.5 for experimental testing purposes in this work, however, this can be adjusted and tailored to fit any implementation domain based on the need of that domain. For example, a highly sensitive domain may choose to reduce the threshold, such that more query samples are recognised as re-identification attempts. Adopting this in the research experiment with a new threshold of 0.4 for instance, will increase the *Tokenizer* MLP unseen test performance, as attack Sample 2 from the NPD will then be classified as a re-identification attempt. The *Tokenizer* KNN classifier will improve in unseen data test performance, as three more attack samples (1, 2, and 6) from the NPD database will be recognised as attack attempts and two more in the *TF-IDF* Euclidean KNN. This will simultaneously increase the FPR in the *Tokenizer* KNN classifier, as normal samples 1, 3, 5, and 7 from the GIC database will be considered attack queries by the classifier. This change in the threshold will not affect predictions in the LR classifier. The *Tokenizer* MNB classifier will have to predict all samples (attack and normal) to be attacking instances at a lower threshold value, however, the *TF-IDF* MNB will improve in performance as attack sample 3 from the NPD database will be classified as an attacking sample. A less sensitive domain can choose to increase the threshold for more leniency in the system. Based on the prediction scores, the implementation domain can choose a different course of action to address the data privacy threat posed by the re-identification attempt, similar to how any query with a score between 6.0 and 8.0 is automatically blocked in [78].

The *Tokenizer* KNN classifier produced results similar to the MLP and LR in the prediction of unseen samples. Attack samples 4, 5, and 7 from the NPD database, and attack samples 1, 3, 4, 5, 6, and 7 from the GIC database are predicted as attack queries. The recognition of attacking features in samples 4, 5, and 7 from the two databases by the MLP, LR, and KNN classifier

further support the inference that there may be a correlation between the attacker’s skill set and the certainty level in the classifier’s recognition of a query sample as a re-identification attack. The performance evaluation metrics achieved in the KNN with Euclidean distance function and the results from the unseen data test are a good indication of generalisation ability and real-life practicability of the KNN classifier for re-identification attack attempt prediction. The KNN classifier has one false positive prediction with Sample 4 from the GIC database, unlike the MLP and LR with no false positives in the unseen data test.

The *Tokenizer* MNB classifier shows a slight consistency with the other classifiers, recognising attack samples 4 and 5 in the NPD database with a higher prediction score. However, there is a high FPR in the unseen data test result for MNB. This shows a low generalisation ability in the classifier. The *Tokenizer* MNB test result indicates inferior real-life practicability compared to the MLP, KNN, and LR models. The *TF-IDF* MNB however, showed a significantly improved generalisation performance and consistency with the MLP and LR classifiers.

The comparisons of unseen data test results for the four classifiers are presented in tables 6.20 and 6.21 below.

Attack Samples	MLP		MNB		KNN		LR	
	NPD	GIC	NPD	GIC	NPD	GIC	NPD	GIC
Sample 1	0.743	0.969	0.54	0.534	0.4	1.0	0.999	1.0
Sample 2	0.450	0.234	0.569	0.503	0.4	0.0	0.026	0.0003
Sample 3	0.129	0.174	0.544	0.488	0.2	0.6	0.695	0.000
Sample 4	0.999	0.989	0.728	0.567	1.0	1.0	1.0	1.0
Sample 5	0.994	0.967	0.666	0.537	1.0	1.0	0.999	1.0
Sample 6	0.794	0.989	0.527	0.539	1.0	1.0	1.0	1.0
Sample 7	0.997	0.999	0.575	0.568	0.2	1.0	1.0	1.0
Normal Samples								
Sample 1	0.001	0.045	0.517	0.486	0.0	0.4	0.000	0.000
Sample 2	0.003	0.083	0.513	0.507	0.2	0.2	0.000	0.000
Sample 3	2.6e-6	0.049	0.507	0.482	0.0	0.4	0.000	0.000
Sample 4	4.3e-6	0.089	0.519	0.495	0.0	0.6	0.000	0.000
Sample 5	0.0004	0.083	0.502	0.488	0.0	0.4	0.000	0.000
Sample 6	0.0005	0.146	0.457	0.512	0.2	1.0	0.000	0.000
Sample 7	0.002	0.076	0.538	0.488	0.4	1.0	0.000	0.000

Table 6.20: Comparison of unseen data test results for MLP, MNB, KNN and LR classifiers (*Tokenizer*)

Attack Samples	MLP		MNB		KNN		LR	
	NPD	GIC	NPD	GIC	NPD	GIC	NPD	GIC
Sample 1	0.999	0.999	0.959	0.984	0.2	0.4	1.0	1.0
Sample 2	0.756	0.998	0.379	0.005	0.0	0.0	0.000	1.0
Sample 3	0.000	0.005	0.458	0.180	0.0	0.6	0.000	0.000
Sample 4	1.0	0.999	0.999	0.991	1.0	0.6	1.0	1.0
Sample 5	1.0	0.999	0.999	0.985	1.0	0.4	1.0	1.0
Sample 6	0.000	0.999	0.556	0.966	0.8	1.0	0.000	1.0
Sample 7	0.994	1.0	0.985	0.998	1.0	1.0	1.0	1.0
Normal Samples								
Sample 1	0.000	0.000	0.009	0.157	0.2	0.6	0.000	0.000
Sample 2	0.000	0.0001	0.006	0.017	0.0	1.0	0.000	0.000
Sample 3	0.000	0.000	0.274	0.156	0.0	0.8	0.000	0.000
Sample 4	0.000	0.0007	0.430	0.202	0.2	0.8	0.000	0.000
Sample 5	0.0004	0.0004	0.051	0.188	0.0	0.6	0.000	0.000
Sample 6	0.0005	0.0006	0.001	0.016	0.0	1.0	0.000	0.000
Sample 7	0.002	0.0001	0.001	0.189	0.4	0.6	0.000	0.000

Table 6.21: Comparison of unseen data test results for MLP, MNB, KNN and LR classifiers (*TF-IDF*)

Overall, the MLP with *TF-IDF* features, tuned MNB with *TF-IDF*, KNN with Euclidean distance metric using *Tokenizer* features, and tuned LR with *Tokenizer* are the best-performing classifier models for recognising re-identification attack attempts as a series of SQL queries on an anonymised database.

6.6 Chapter Summary

This chapter presents the performance evaluation of the machine learning algorithms explored for the recognition of re-identification patterns in SQL queries on anonymised datasets. The performance evaluation results for the different models trained and tested against the testing set are presented and compared using performance evaluation metrics including accuracy, precision, recall, f-score, error rate, true negative rate, false positive rate, and AUC. Using the features extracted from the research dataset with both the *Tokenizer* and *TF-IDF* feature extraction techniques, MLP, baseline and tuned MNB, KNN with both Euclidean and Manhattan distance functions, and baseline and tuned LR are all evaluated with the testing dataset using these metrics. The evaluation showed that the tuned versions of MNB and LR are better in the classification performance. KNN with Manhattan distance function showed better performance evaluation, however, the Euclidean distance function trained KNN showed better generalisation ability when tested with the unseen dataset.

The unseen data test results showed that the MLP trained with *TF-IDF* features, tuned MNB with *TF-IDF* and LR with *Tokenizer* features have the best generalisation capabilities, recognising and predicting most of the previously unseen re-identification attempt queries as re-identification attack attempts. These classifiers also show a 0% FPR, the MLP recognises re-identification attempts 79% of the time, MNB correctly predicts unseen attack samples with 71% accuracy and the LR model does the same in 79% of unseen query instances, indicating a

real-world practical application of the classifiers. The *Tokenizer* KNN with Euclidean distance also shows generalisation skills that indicate practicability. The performance evaluation metrics scores and the unseen data test result in the *Tokenizer* MNB suggest that there is overfitting in the model, as the model shows 26% FPR during evaluation and predicted 8 of 14 unseen normal queries as re-identification attempts.

Chapter 7

Conclusion and Recommendation

Four classification algorithms are explored in this research work for the recognition and prediction of data re-identification attack attempts on anonymised databases, as demonstrated with the unseen data test in section 6.4. The classifier models are trained to recognise patterns in SQL queries that are executed on anonymised databases that may suggest that the users on such databases are performing re-identification. Four algorithms: multilayer perceptron, naive Bayes, k-nearest neighbors, and logistic regression are implemented, as described and justified in chapter 3. In section 7.1, this chapter presents an overall description of how the research aim and objectives outlined in this work are addressed. The original academic contributions achieved in this research work are discussed in 7.2 and section 7.3 provides insights related to the different related areas that could be explored for further research in the future, to consolidate and advance the achievements in this work.

7.1 Introduction

With the vast increase in the amount of data being collected, stored, and shared about individuals in the modern technological age, data anonymisation is used by data holders to maintain the privacy of the data subjects. Databases are shared for many reasons, including data processing for the advancement of knowledge and generation of meaningful information. Different data anonymisation criteria are implemented by data holders in attempts to keep PII contained in the shared databases safe from unauthorised access. However, the threat posed by data re-identification attacks is one that can not be ignored. This is emphasised in various literature analysed in chapter 2 of this thesis. Data re-identification uses publicly accessible sources of information to link a subject or multiple subjects in an anonymised database to their personal and sensitive information, information believed to be privacy-safe because of the anonymisation criteria applied to it before sharing the database. As it pertains to data sharing, relational DBMS that stores anonymised datasets do not provide security measures against an attack and data misuse such as re-identification. Real cases of successful re-identification scenarios explored in this research showed that re-identification is achievable by combining the anonymised database with other sources of information. The issue of re-identification of anonymised datasets has been addressed in different literature, with the majority of the work suggesting better anonymisation and contextual risk assessment as countermeasures against data re-identification attacks.

Information stored on databases is accessed and manipulated with the use of SQL queries,

and the presupposition in this research is that the re-identification of anonymised databases can result from a series of SQL queries executed against such databases. In chapter 4, a re-identification attack re-creation experiment is conducted, whereby a re-identification attack is successfully performed on anonymised databases. Two different synthetic databases are explored for the re-identification experiment in this work. The synthetic databases are modelled after real-life cases of successful re-identification attacks. These are the Netflix Prize Data re-identification scenario and the re-identification of Governor William Weld from the GIC database. In both cases, re-identification was accomplished with the use of a secondary database to link the information of the subjects of the attacks. As proposed by this research in section 2.4.1, there are three different objectives re-identification attacks can accomplish, based on the motive of the attacker. The re-identification attack in the NPD is identified to have an existential objective and the GIC database re-identification is a targeted attack. The success of the research experiment to represent re-identification attacks as a series of SQL queries sets the basis for a data mining exploration to recognise patterns in SQL queries executed on anonymised databases that constitute a re-identification attack. The research problem to recognise re-identification patterns in database queries is explored as a supervised, binary classification task with a class belonging to queries attempting re-identification attacks (abnormal) and another class for non re-identification (normal) database queries. Data collection campaign to gather SQL query logs for each of the classes was launched. The data collection process is described in section 4.6.

Different works from academic literature implementing data mining techniques and machine learning algorithms to analyse SQL query logs are analysed. The SQL query logs analysis is used for solving a variety of problems, including anomalous pattern recognition and automated intrusion detection system. However, to the best of this research's knowledge, none of the available existing work has analysed SQL query logs for re-identification query pattern recognition. A review of machine learning algorithms implemented in pattern recognition and binary classification is presented and justification for selecting the algorithms is also provided. The data mining experiment in this research uses data science libraries available in the Python programming language to implement the machine learning algorithms applied to the research problem of learning re-identification attack patterns in SQL queries. The SQL query log data collected for this research is used to train different classification algorithms to learn re-identification patterns from the research dataset. This research uses hyperparameter tuning to achieve better performance in machine learning models and a new set of unseen data is gathered to test the generalisation ability of the classification models developed. The multilayer perceptron, k-nearest neighbors, and logistic regression achieved high metrics scores during performance evaluation. The multilayer perceptron, tuned multinomial Naive Bayes, and logistic regression are found to have a practicable generalisation ability during unseen data tests. The other classifiers have scores that are relatively lower and or show unseen data test results that indicate mediocre generalisation ability.

7.2 Contributions

The main contribution of this research work is establishing that the re-identification of subject(s) in an anonymised database is possible through a series of SQL queries executed on a database, and the development of novel classification models that can recognise patterns in SQL queries that may be attempting re-identification attacks on an anonymised database. The developed classifiers are unique in their ability to recognise and distinguish between re-identification attack

attempt patterns in SQL queries and normal (non re-identification) query patterns. An overview of the main contributions of this research is presented in this section. A summary of the methodical approach taken to achieve the research objectives and answer the research questions (RQ) is depicted in Figure 7.1 below. The figure describes the different phases in the research process, from the literature review to the evaluation and testing of the developed models for re-identification attack attempts recognition.

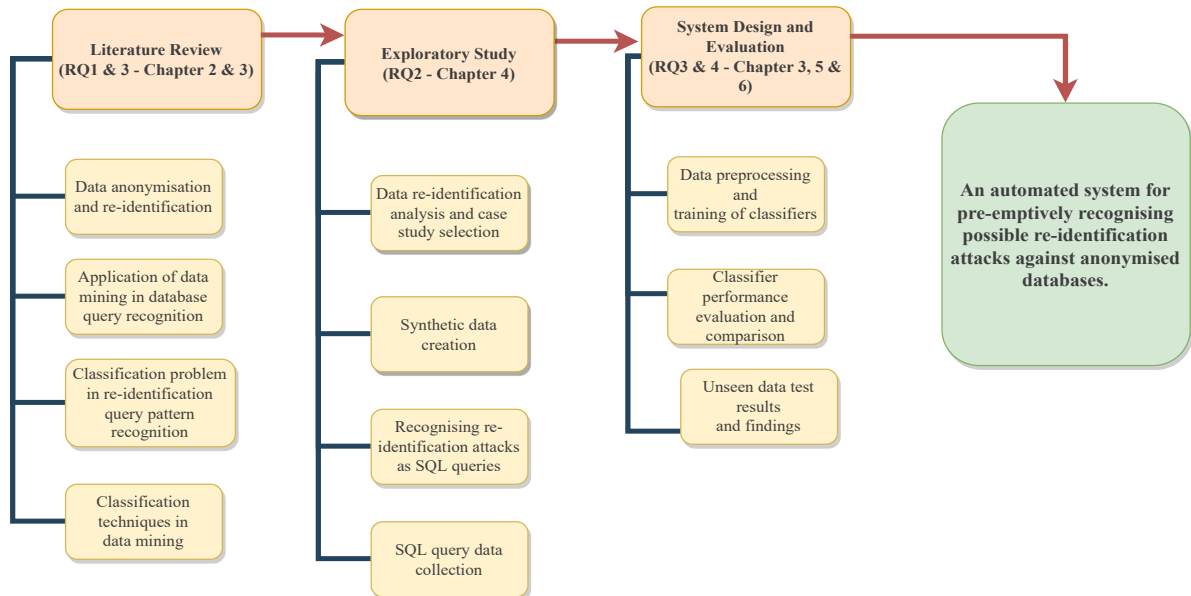


Figure 7.1: Research Phases and Contributions

1. **Literature review:** Chapter 2 of this thesis presented an overview data privacy threats posed by re-identification attacks against anonymised databases, as reported in the literature reviewed in the chapter. The analysis of the literature shows that the implementation of anonymisation criteria on databases is not sufficient to keep personal and sensitive information about subjects of an anonymised database safe, particularly in the context of data sharing. The chapter draws a parallel in the perspective that the reviewed publications are approaching anonymisation and re-identification. The proposed solutions to the threat of re-identification by the reviewed publications are also presented. Discussions surrounding the different concepts in anonymisation and re-identification are detailed. A review of published work related to the application of data mining for database query recognition is explored in chapter 3. The literature review exploration addresses research objective 1 outlined in section 1.5 and research questions 1 and 3 as outlined in section 1.3.
2. **Problem statement and justification of algorithms:** An overview of different data mining techniques is presented in chapter 3. The nature of the research problem is analysed and perceived to be a binary classification problem. Analysis of different relevant binary classification algorithms to predict the re-identification attack likelihood in a series of database queries is presented and justified. The analysis in chapter 3 addresses research objectives 1 and 2 as well as answers research questions 1 and 3.
3. **Exploratory study:** A main contribution to knowledge as a result of this research is the work covered in chapter 4. In this chapter, an exploratory study is conducted to ascertain

that re-identification attacks can be in the form of a series of SQL database queries and that these queries have enough in common to constitute a recognisable pattern. The approach employed in achieving this objective involved studying real cases of re-identification attacks, and using the gathered information to build a synthetic anonymised dataset that mimics the structure of the dataset in the re-identification cases. Netflix Prize Dataset and the re-identification of Governor William Weld from the GIC dataset are used as case studies to build the synthetic dataset for research. Using SQL queries, re-identification attacks are attempted on these anonymised synthetic databases, to establish that re-identification can occur through a series of database queries. No known work has been identified in this area to explore what constitutes re-identification attacks from a technical perspective. The work done at this stage of the research addresses research objective 2 and research question 2.

4. **Re-identification query pattern recognition:** Another novel contribution from this work is the development of data mining models for automatic recognition of re-identification attack query patterns on anonymised databases, using classification algorithms. The experiment conducted to achieve this is detailed in chapters 5 and 6. This research explored MLP, naive Bayes, KNN, and logistic regression algorithms for the training of classifier models to learn re-identification patterns in SQL query datasets. The model training was conducted as supervised learning with labelled datasets (normal and re-identification attack attempt queries), and these four algorithms are implemented due to their application in pattern recognition and binary classification problems. The MLP, MNB, and logistic regression models showed the best performance in their re-identification query pattern recognition. The MLP achieved 100% accuracy during evaluation and is able to correctly predict attack attempts 79% of the time when tested with unseen datasets. The accuracy for the most applicable MNB model is recorded to be 79.3% and correctly classifies unseen attack samples 71% of the time. The LR classifier achieved an accuracy of 100% during performance evaluation and recognises 79% of re-identification attack queries during unseen data testing. This phase of the research addresses research objective 3 and research questions 3 and 4.
5. **Potential application area:** As mentioned in the discussion about the background of this research in section 1.1, platforms such as NHS Digital TRE service for England can potentially integrate the function of the classifiers developed by this research to access SQL queries executed by researchers using the Data Access Environment for re-identification attack attempts. The integration of an automated model that predicts re-identification likelihood productively contributes to the maintenance of privacy-safe access to health and personal data sharing that the TRE service aims to accomplish.
6. **Re-identification query dataset:** The classifiers developed in this work are novel systems trained with original datasets. These datasets are gathered as part of the research methods implemented in this work. The datasets are a contribution to academic knowledge, as other researchers can use them in their work. This research plans to publish the SQL query dataset collected and used in training and testing the classifiers in the research data mining experiment.

The work in this research and the results generated put forward positive contributions to the data privacy domain, as it relates to anonymisation, re-identification, and data sharing. The exploratory study conducted in Chapter 4 was accepted for a conference presentation and publication, after being peer-reviewed. The work done in the study creates the basis for achieving

the aim set for the research. There are current efforts to produce and submit more academic papers for publication, relating to the results achieved in this research as presented in Chapter 6.

7.3 Future Work

This research has created a state-of-the-art model for predicting re-identification attack attempts on anonymised databases. However, there are avenues related to this research work where further exploration could be done to advance the achievements of this research. While the objectives of this research have been satisfied, the outcome of the data mining experiment can be further improved given access to a large amount of *normal* and *re-identification attempt* SQL query samples. The methodology devised in this research (proven to be effective based on the test results) can then be used to train classifiers. With a substantial amount of SQL query samples to train models about re-identification and non re-identification patterns, classifier performance can be improved further, especially generalisation performance.

Moreover, this research postulates that re-identification attacks could have three objectives based on the motive of the attacker, as discussed in section 2.4.1. A data mining experiment can be conducted to learn the pattern in re-identification attacks based on their objectives. Future work in this direction will be required to generate a SQL query profile of what re-identification attacks entail with a specific objective to be achieved by the attacker. Machine learning algorithms can then be trained to predict not only that a query sample is re-identification prone, but what the objective of the attack is. Also, the work done in this research shows that there may be re-identification attack cases that cannot be technically represented with database search queries, as described in section 4.3.3. Exploration into these re-identification case studies with the aim of understanding what they entail and how to interpret them technically will be a productive contribution in this domain.

7.4 Chapter Summary

The overall aim and objectives in this research as stated in section 1.5 have been satisfied. The initial questions posed by the research have been answered with work presented across the different chapters in this thesis. This chapter started by reiterating the background, literature review, methodology, experiments, and results of this research. The original contributions to academic knowledge as a result of this research are enumerated in section 7.2, including the review of the literature to find the gap in the existing measures against re-identification attacks, an exploratory study to ascertain that re-identification can be achieved through a series of SQL queries on an anonymised database and developing automated system to recognise and predict the likelihood of re-identification in database queries. The aim of the research was successful, with three models from this research (MLP, MNB and LR) showing performance that is practicable and deployable in a real-world environment. As mentioned in this chapter, the product of this research has potential application in areas such as the NHS Trusted Research Environment where anonymised health data is shared with researchers for analysis. This research has one related publication, with more papers in development for more publications, to share the work and results achieved in this thesis with the wider academic community.

References

- [1] The keys to data protection. Technical report, Privacy International, 62 Britton Street, London, UK, 2018.
- [2] Herve Abdi. A neural network primer. *Journal of Biological Systems*, 2(03):247–281, 1994.
- [3] Esma Aïmeur, Gilles Brassard, and Paul Molins. Reconstructing profiles from information disseminated on the internet. In *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Conference on Social Computing*, pages 875–883. IEEE, 2012.
- [4] Mohamed Aly. Survey on multiclass classification methods. *Neural Netw*, 19:1–9, 2005.
- [5] Ateret Anaby-Tavor, Boaz Carmeli, Esther Goldbraich, Amir Kantor, George Kour, Segev Shlomov, Naama Tepper, and Naama Zwerdling. Do not have enough data? deep learning to the rescue! In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7383–7390, 2020.
- [6] Jun Chin Ang, Andri Mirzal, Habibollah Haron, and Haza Nuzly Abdull Hamed. Supervised, unsupervised, and semi-supervised feature selection: a review on gene selection. *IEEE/ACM transactions on computational biology and bioinformatics*, 13(5):971–989, 2015.
- [7] Olivia Angiuli, Joe Blitzstein, and Jim Waldo. How to de-identify your data. *Communications of the ACM*, 58(12):48–55, 2015.
- [8] Nari S Arunraj, Robert Hable, Michael Fernandes, Karl Leidl, and Michael Heigl. Comparison of supervised, semi-supervised and unsupervised learning methods in network intrusion detection system (nids) application. *Anwendungen und Konzepte der Wirtschaftsinformatik*, 6:10–19, 2017.
- [9] Daniel Barth-Jones. The re-identification of governor william weld’s medical information: a critical re-examination of health data identification risks and privacy protections, then and now. 2012.
- [10] Youssef Bassil. A comparative study on the performance of the top dbms systems. *arXiv preprint arXiv:1205.2889*, 2012.
- [11] Markus Bayer, Marc-André Kaufhold, and Christian Reuter. A survey on data augmentation for text classification. *ACM Computing Surveys*, 55(7):1–39, 2022.
- [12] Colin Bellinger, Shiven Sharma, and Nathalie Japkowicz. One-class versus binary classification: Which and when? In *2012 11th International Conference on Machine Learning and Applications*, volume 2, pages 102–106. IEEE, 2012.

- [13] Daniel Berrar. Cross-validation., 2019.
- [14] Ritesh Bhagwat, Mahla Abdollahnejad, and Matthew Moocarme. *Applied deep learning with keras: Solve complex real-life problems with the simplicity of keras*. Packt Publishing Ltd, 2019.
- [15] Mrs Bharati and M Ramageri. Data mining techniques and applications. *Indian Journal of Computer Science and Engineering*, 1(4):301–305, 2010.
- [16] Neeraj Bhargava, Atif Aziz, and Rajiv Arya. Selection criteria for data mining software: A study. *International Journal of Computer Science Issues (IJCSI)*, 10(3):308, 2013.
- [17] Eerke Boiten. Nhs plan to share gp patient data postponed – but will new measures address concerns? 2021. [Online]. Available: <https://theconversation.com/nhs-plan-to-share-gp-patient-data-postponed-but-will-new-measures-address-concerns-165103>.
- [18] Luca Bolognini and Camilla Bistolfi. Pseudonymization and impacts of big (personal/anonymous) data processing in the transition from the directive 95/46/ec to the new eu general data protection regulation. *Computer law & security review*, 33(2):171–181, 2017.
- [19] Andrew P Bradley. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern recognition*, 30(7):1145–1159, 1997.
- [20] Jason Brownlee. *Deep learning for natural language processing: develop deep learning models for your natural language problems*. Machine Learning Mastery, 2017.
- [21] Jason Brownlee. *Better deep learning: train faster, reduce overfitting, and make better predictions*. Machine Learning Mastery, 2018.
- [22] Ji-Won Byun, Ashish Kamra, Elisa Bertino, and Ninghui Li. Efficient k-anonymization using clustering techniques. In *International Conference on Database Systems for Advanced Applications*, pages 188–200. Springer, 2007.
- [23] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):1–58, 2009.
- [24] Nitesh V Chawla, Nathalie Japkowicz, and Aleksander Kotcz. Special issue on learning from imbalanced data sets. *ACM SIGKDD explorations newsletter*, 6(1):1–6, 2004.
- [25] Valentina Ciriani, S De Capitani Di Vimercati, Sara Foresti, and Pierangela Samarati. κ -anonymity. In *Secure data management in decentralized systems*, pages 323–353. Springer, 2007.
- [26] Chris Clifton and Tamir Tassa. On syntactic anonymity and differential privacy. In *2013 IEEE 29th International Conference on Data Engineering Workshops (ICDEW)*, pages 88–93. IEEE, 2013.
- [27] Ken Collier, Bernard Carey, Donald Sautter, and Curt Marjaniemi. A methodology for evaluating and selecting data mining software. In *Proceedings of the 32nd Annual Hawaii International Conference on Systems Sciences. 1999. HICSS-32. Abstracts and CD-ROM of Full Papers*, pages 11–pp. IEEE, 1999.
- [28] Cornelius Crowley. Setting standards for microdata. *ECB Conference on Statistics*, 2016.

-
- [29] Tore Dalenius. Finding a needle in a haystack or identifying anonymous census records. *Journal of Official Statistics*, 2:329–336, 1986.
- [30] Fida Kamal Dankar and Khaled El Emam. Practicing differential privacy in health care: A review. *Trans. Data Priv.*, 6(1):35–67, 2013.
- [31] Fortunato S De Menezes, Gilberto R Liska, Marcelo A Cirillo, and Mário JF Vivanco. Data classification with binary response through the boosting algorithm and logistic regression. *Expert Systems with Applications*, 69:62–73, 2017.
- [32] Yves-Alexandre De Montjoye, César A Hidalgo, Michel Verleysen, and Vincent D Blondel. Unique in the crowd: The privacy bounds of human mobility. *Scientific reports*, 3(1):1–5, 2013.
- [33] Yves-Alexandre De Montjoye, Laura Radaelli, Vivek Kumar Singh, et al. Unique in the shopping mall: On the reidentifiability of credit card metadata. *Science*, 347(6221):536–539, 2015.
- [34] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- [35] Lopamudra Dey, Sanjay Chakraborty, Anuraag Biswas, Beepa Bose, and Sweta Tiwari. Sentiment analysis of review datasets using naive bayes and k-nn classifier. *arXiv preprint arXiv:1610.09982*, 2016.
- [36] Tom Dietterich. Overfitting and undercomputing in machine learning. *ACM computing surveys (CSUR)*, 27(3):326–327, 1995.
- [37] NHS Digital. Trusted research environment service for england. Online, 2022. Available: <https://digital.nhs.uk/coronavirus/coronavirus-data-services-updates/trusted-research-environment-service-for-england>.
- [38] NHS Digital. Hospital episode statistics (hes). Online, 2022. Available: <https://digital.nhs.uk/data-and-information/data-tools-and-services/data-services/hospital-episode-statistics>.
- [39] Neslihan Dogan and Zuhul Tanrikulu. A comparative analysis of classification algorithms in data mining for accuracy, speed and robustness. *Information Technology and Management*, 14(2):105–124, 2013.
- [40] Josep Domingo-Ferrer and Vicenç Torra. A critique of k-anonymity and some of its enhancements. In *2008 Third International Conference on Availability, Reliability and Security*, pages 990–993. IEEE, 2008.
- [41] Stephan Dreiseitl and Lucila Ohno-Machado. Logistic regression and artificial neural network classification models: a methodology review. *Journal of biomedical informatics*, 35(5-6):352–359, 2002.
- [42] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of cryptography conference*, pages 265–284. Springer, 2006.
-

- [43] Cynthia Dwork, Aaron Roth, et al. The algorithmic foundations of differential privacy. *Foundations and Trends in Theoretical Computer Science*, 9(3-4):211–407, 2014.
- [44] IBM Cloud Education. Neural networks. Online, 2020. [Accessed 20/04/2022] Available: <https://www.ibm.com/cloud/learn/neural-networks>.
- [45] IBM Cloud Education. Unsupervised learning. Online, 2020. [Accessed 20/05/2022] Available: <https://www.ibm.com/cloud/learn/unsupervised-learning>.
- [46] IBM Cloud Education. Convolutional neural networks. 2020 [Accessed 23/04/2022] Available: <https://www.ibm.com/cloud/learn/convolutional-neural-networks>.
- [47] IBM Cloud Education. Recurrent neural networks. 2020 [Accessed 23/04/2022] Available: <https://www.ibm.com/cloud/learn/recurrent-neural-networks>.
- [48] IBM Cloud Education. Gradient descent [online]. 2020 [Accessed 30/04/2022] Available: <https://www.ibm.com/cloud/learn/gradient-descent>.
- [49] Mark Elliot, Elaine Mackey, Kieron O’Hara, and Caroline Tudor. *The anonymisation decision-making framework*. UKAN Manchester, 2016.
- [50] Eurostat. How to apply for microdata? *European Commission*, 2017.
- [51] Victor AE Farias, José GR Maia, Flávio RC Sousa, Leonardo O Moreira, Gustavo AC Campos, and Javam C Machado. A machine learning approach for sql queries response time estimation in the cloud. *Proceedings of the Simpósio Brasileiro de Banco de Dados (SBBD)*, pages 1–6, 2013.
- [52] Dina Fawzy, Sherin Moussa, and Nagwa Badr. The evolution of data mining techniques to big data analytics: An extensive study with application to renewable energy data analytics. *Asian Journal of Applied Sciences*, 4(3), 2016.
- [53] Jiashi Feng, Huan Xu, Shie Mannor, and Shuicheng Yan. Robust logistic regression and classification. *Advances in neural information processing systems*, 27, 2014.
- [54] César Ferri, José Hernández-Orallo, and R Modroiu. An experimental comparison of performance measures for classification. *Pattern recognition letters*, 30(1):27–38, 2009.
- [55] Simson L. Garfinkel. De-identification of personal information. *NIST Interagency/Internal Report (NISTIR)*, 8053, 2015.
- [56] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. ” O’Reilly Media, Inc.”, 2019.
- [57] Gabriel Ghinita, Panagiotis Karras, Panos Kalnis, and Nikos Mamoulis. A framework for efficient data anonymization under privacy and accuracy constraints. *ACM Trans. Database Syst.*, 34(2):9:1–9:47, July 2009.
- [58] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.
- [59] Gov.uk. Data protection act 2018. *UK Public General Acts*, 2018. <http://www.legislation.gov.uk/ukpga/2018/12/enacted/data.pdf>.

-
- [60] Qiong Gu, Li Zhu, and Zhihua Cai. Evaluation measures of the classification performance of imbalanced data sets. In *International symposium on intelligence computation and applications*, pages 461–471. Springer, 2009.
- [61] Antonio Gulli and Sujit Pal. *Deep learning with Keras*. Packt Publishing Ltd, 2017.
- [62] Jiawei Han, Micheline Kamber, and Jian Pei. Data mining concepts and techniques third edition. *The Morgan Kaufmann Series in Data Management Systems*, 5(4):83–124, 2011.
- [63] Muhammad Hasnain, Muhammad Fermi Pasha, Imran Ghani, Muhammad Imran, Mohammed Y Alzahrani, and Rahmat Budiarto. Evaluating trust prediction and confusion matrix measures for web services ranking. *IEEE Access*, 8:90847–90861, 2020.
- [64] M Hossin, MN Sulaiman, A Mustapha, N Mustapha, and RW Rahmat. A hybrid evaluation metric for optimizing classifier. In *2011 3rd Conference on Data Mining and Optimization (DMO)*, pages 165–170. IEEE, 2011.
- [65] Mohammad Hossin and Md Nasir Sulaiman. A review on evaluation metrics for data classification evaluations. *International journal of data mining & knowledge management process*, 5(2):1, 2015.
- [66] Jin Huang and Charles X Ling. Using auc and accuracy in evaluating learning algorithms. *IEEE Transactions on knowledge and Data Engineering*, 17(3):299–310, 2005.
- [67] Viet H Huynh and An NT Le. Process mining and security: visualization in database intrusion detection. In *Pacific-Asia Workshop on Intelligence and Security Informatics*, pages 81–95. Springer, 2012.
- [68] IBM. Supervised learning. Online, 2020. [Accessed 16/03/2021] Available: <https://www.ibm.com/cloud/learn/supervised-learning>.
- [69] IBM. What is logistic regression? Online, [Accessed 20/05/2022] Available: <https://www.ibm.com/uk-en/topics/logistic-regression>.
- [70] ICO. Anonymisation: managing data protection risk code of practice. *Information Commissioner’s Office*, 2012. <https://ico.org.uk/media/1061/anonymisation-code.pdf>.
- [71] Miloš Ilić, Lazar Kopanja, Dragan Zlatković, Milica Trajković, and Dejana Čurguz. Microsoft sql server and oracle: Comparative performance analysis. In *The 7th International conference Knowledge management and informatics*, pages 33–40, 2021.
- [72] Information and Privacy Commissioner of Ontario. De-identification guidelines for structured data. *Information and Privacy Commissioner of Ontario*, June 2016.
- [73] Olabayo Ishola. Sql queries. *OneDrive*, [CSV files]. Date Accessed: Oct. 10, 2021. Available: <https://1drv.ms/u/s!ApiLVfIRE8pPmUSMlKvvubOdmYaU?e=1ZOaZM>.
- [74] Olabayo Ishola. Synthetic database for re-identification scenarios. *OneDrive*, [Microsoft Excel spreadsheet files]. Date Accessed: Oct. 10, 2021. Available: <https://1drv.ms/u/s!ApiLVfIRE8pPmT94GdyFB1HvZByp?e=fSM3oH>.
- [75] Olabayo Ishola. Phd experiment, [Video recording]. Date Accessed: Aug. 25, 2021. Available: <https://www.youtube.com/watch?v=GJRi5fI5rqw>.
-

- [76] Olabayo Ishola, Eerke Albert Boiten, Aladdin Ayesha, and Adham Albakri. Recognising re-identification attacks on databases, by interpreting them as sql queries: a technical study. In *Privacy in Statistical Databases (PSD2020)*, Arezzo, Italy, September 2020.
- [77] A Jangra, D Bishla, Komal Bhatia, and Priyanka Priyanka. Functionality and security analysis of oracle, ibm-db2 & sql server. *Global Journal of Computer Science and Technology*, 10(7), 2010.
- [78] Souparnika Jayaprakash and Kamalanathan Kandasamy. Database intrusion detection system using ocraplet and machine learning. In *2018 Second International Conference on Inventive Communication and Computational Technologies (ICICCT)*, pages 1413–1416. IEEE, 2018.
- [79] James Joyce. Bayes’ theorem. 2003.
- [80] Kaggle. Aol user session collection 500k, <https://www.kaggle.com/dineshydv/aol-user-session-collection-500k>. Online, 2006.
- [81] Kaggle. Netflix prize data, <https://www.kaggle.com/netflix-inc/netflix-prize-data>. Online, 2017.
- [82] Hajer Kamel, Dhahir Abdulah, and Jamal M Al-Tuwaijari. Cancer classification using gaussian naive bayes algorithm. In *2019 International Engineering Conference (IEC)*, pages 165–170. IEEE, 2019.
- [83] Ashish Kamra, Evimaria Terzi, and Elisa Bertino. Detecting anomalous access patterns in relational databases. *The VLDB Journal*, 17(5):1063–1077, 2008.
- [84] Aman Kataria and MD Singh. A review of data classification using k-nearest neighbour algorithm. *International Journal of Emerging Technology and Advanced Engineering*, 3(6):354–360, 2013.
- [85] Gopalan Kesavaraj and Sreekumar Sukumaran. A study on classification techniques in data mining. In *2013 fourth international conference on computing, communications and networking technologies (ICCCNT)*, pages 1–7. IEEE, 2013.
- [86] Khushbu Khamar. Short text classification using knn based on distance function. *International Journal of Advanced Research in Computer and Communication Engineering*, 2(4):1916–1919, 2013.
- [87] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [88] Lukasz Kniola. Calculating the risk of re-identification of patient-level data using quantitative approach. *PhUSE*, 2016.
- [89] Pieter Kubben, Michel Dumontier, and Andre Dekker. Fundamentals of clinical data science. 2019.
- [90] Max Kuhn, Kjell Johnson, et al. *Applied predictive modeling*, volume 26. Springer, 2013.
- [91] Merima Kulin, Carolina Fortuna, Eli De Poorter, Dirk Deschrijver, and Ingrid Moerman. Data-driven design of intelligent wireless networks: An overview and tutorial. *Sensors*, 16(6):790, 2016.

-
- [92] Varun Kumar, Ashutosh Choudhary, and Eunah Cho. Data augmentation using pre-trained transformer models. *arXiv preprint arXiv:2003.02245*, 2020.
- [93] Mian Mian Lau and King Hann Lim. Review of adaptive activation function in deep neural network. In *2018 IEEE-EMBS Conference on Biomedical Engineering and Sciences (IECBES)*, pages 686–690. IEEE, 2018.
- [94] legislation.gov.uk. Data protection act 2018. Online, 2018. Available: <https://www.legislation.gov.uk/ukpga/2018/12/contents/enacted/data.pdf>.
- [95] Ninghui Li, Tiancheng Li, and Suresh Venkatasubramanian. t-closeness: Privacy beyond k-anonymity and l-diversity. In *2007 IEEE 23rd International Conference on Data Engineering*, pages 106–115. IEEE, 2007.
- [96] Boris Lubarsky. Re-identification of anonymized data. *Georgetown Law Review*, 202, 2017.
- [97] Son T Luu, Kiet Van Nguyen, and Ngan Luu-Thuy Nguyen. Empirical study of text augmentation on social media text in vietnamese. *arXiv preprint arXiv:2009.12319*, 2020.
- [98] Ashwin Machanavajjhala, Daniel Kifer, Johannes Gehrke, and Muthuramakrishnan Venkatasubramanian. l-diversity: Privacy beyond k-anonymity. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 1(1):3–es, 2007.
- [99] Fathimath Zuha Maksood and Geetha Achuthan. Analysis of data mining techniques and its applications. *International Journal of Computer Applications*, 140(3):6–14, 2016.
- [100] S Mayil and M Vanitha. Data privacy preservation using various perturbation techniques. *International Journal of Advanced Research Trends in Engineering and Technology*, 3, 2016.
- [101] Andrew McCallum, Kamal Nigam, et al. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Citeseer, 1998.
- [102] Vangelis Metsis, Ion Androutsopoulos, and Georgios Paliouras. Spam filtering with naive bayes-which naive bayes? In *CEAS*, volume 17, pages 28–69. Mountain View, CA, 2006.
- [103] Microsoft. Download sql server management studio (ssms). *Microsoft Docs*, Date Accessed: Aug. 27, 2021. [Online]. Available: <https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms?view=sql-server-ver15>.
- [104] Microsoft. Sql server profiler. *Microsoft Docs*, Date Accessed: Aug. 27, 2021. [Online]. Available: <https://docs.microsoft.com/en-us/sql/tools/sql-server-profiler/sql-server-profiler?view=sql-server-ver15>.
- [105] Kato Mivule. Utilizing noise addition for data privacy, an overview. *arXiv preprint arXiv:1309.3958*, 2013.
- [106] Shafayat Bin Shabbir Mugdha, Marian Binte Mohammed Mainuddin Kuddus, Lubaba Salsabil, Afra Anika, Piya Prue Marma, Zahid Hossain, and Swakkhar Shatabda. A gaussian naive bayesian classifier for fake news detection in bengali. In *Emerging Technologies in Data Mining and Information Security*, pages 283–291. Springer, 2021.
- [107] A. Narayanan and V. Shmatikov. Robust de-anonymization of large sparse datasets. In *2008 IEEE Symposium on Security and Privacy (sp 2008)*, pages 111–125, May 2008.
-

- [108] Gregory S Nelson. Practical implications of sharing data: a primer on data privacy, anonymization, and de-identification. In *SAS Global Forum Proceedings*, pages 1–23, 2015.
- [109] Salvador Ochoa, Jamie Rasmussen, Christine Robson, and Michael Salib. Reidentification of individuals in Chicago’s homicide database: A technical and legal study. *Massachusetts Institute of Technology*, 08 2002.
- [110] Paul Ohm. Broken promises of privacy: Responding to the surprising failure of anonymization. *UCLA Law Review*, 57(1701):9–12, August 2010.
- [111] Emilio Soria Olivas, Jos David Mart Guerrero, Marcelino Martinez-Sober, Jose Rafael Magdalena-Benedito, L Serrano, et al. *Handbook of research on machine learning applications and trends: Algorithms, methods, and techniques: Algorithms, methods, and techniques*. IGI Global, 2009.
- [112] P. Priyadharsini P. Dhamodran and M.S. Kavitha. A survey on data anonymization techniques for large data sets. *IJCSMC*, 3(11):144–148, November 2014.
- [113] V Mohan Patro and Manas Ranjan Patra. Augmenting weighted average with confusion matrix to enhance classification accuracy. *Transactions on Machine Learning and Artificial Intelligence*, 2(4):77–91, 2014.
- [114] Pramuditha Perera and Vishal M Patel. Learning deep features for one-class classification. *IEEE Transactions on Image Processing*, 28(11):5450–5463, 2019.
- [115] Mark Phillips, Edward S Dove, and Bartha M Knoppers. Criminal prohibition of wrongful re-identification: Legal solution or minefield for big data? *Journal of bioethical inquiry*, 14(4):527–539, 2017.
- [116] VS Prasatha, Haneen Arafat Abu Alfeilate, AB Hassanate, Omar Lasassmehe, Ahmad S Tarawnehf, Mahmoud Bashir Alhasanatg, and Hamzeh S Eyal Salmane. Effects of distance measure choice on knn classifier performance-a review. *arXiv preprint arXiv:1708.04321*, page 56, 2017.
- [117] N. Punitha and R. Amsaveni. Methods and techniques to protect the privacy information in privacy preservation data mining. *International Journal of Computer Technology and Applications (IJCTA)*, 2(6), 2011.
- [118] Python. The python standard library. *Python 3.10.4 documentation*, Date Accessed: Mar. 30, 2022. Available: <https://docs.python.org/3/library/>.
- [119] Praveen Joe I. R. and Varalakshmi P. A survey on neural network models for data analysis. *ARPJ Journal of Engineering and Applied Sciences*, 10(11), 2015.
- [120] Balaji Raghunathan. *The complete book of data anonymization: from planning to implementation*. CRC Press, 2013.
- [121] Hassan Ramchoun, Youssef Ghanou, Mohamed Ettaouil, and Mohammed Amine Janati Idrissi. Multilayer perceptron: Architecture optimization and training. 2016.
- [122] Romesh Ranawana and Vasile Palade. Optimized precision-a new measure for classifier performance evaluation. In *2006 IEEE International Conference on Evolutionary Computation*, pages 2254–2261. IEEE, 2006.

-
- [123] KV Ranjitha et al. Classification and optimization scheme for text data using machine learning naïve bayes classifier. In *2018 IEEE world symposium on communication engineering (WSCE)*, pages 33–36. IEEE, 2018.
- [124] Mohammad Amin Rashid and Hossein Amirkhani. The effect of using masked language models in random textual data augmentation. In *2021 26th International Computer Conference, Computer Society of Iran (CSICC)*, pages 1–5. IEEE, 2021.
- [125] David Rebollo-Monedero, Jordi Forne, and Josep Domingo-Ferrer. From t-closeness-like privacy to postrandomization via information theory. *IEEE Transactions on Knowledge and Data Engineering*, 22(11):1623–1636, 2009.
- [126] Russell Reed and Robert J MarksII. *Neural smithing: supervised learning in feedforward artificial neural networks*. Mit Press, 1999.
- [127] Sérgio Luís Ribeiro and Emilio Tissato Nakamura. Privacy protection with pseudonymization and anonymization in a health iot system: Results from ocariot. In *2019 IEEE 19th International Conference on Bioinformatics and Bioengineering (BIBE)*, pages 904–908, 2019.
- [128] Luc Rocher, Julien Hendrickx, and Yves-Alexandre de Montjoye. Estimating the success of re-identifications in incomplete datasets using generative models. *Nature Communications*, 10(3069), 2019. <https://cpg.doc.ic.ac.uk/individual-risk/>.
- [129] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.
- [130] Pierangela Samarati and Latanya Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression. 1998.
- [131] Murat H Sazli. A brief review of feed-forward neural networks. *Communications Faculty of Sciences University of Ankara Series A2-A3 Physical Sciences and Engineering*, 50(01), 2006.
- [132] Scikit-learn. Kneighnorsclassifier. Online, [Accessed 08/05/2022] Available: <https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html>.
- [133] Scikit-learn. Logisticregression. Online, [Accessed 10/05/2022] Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html.
- [134] Scikit-learn. Linear models. Online, [Accessed 10/05/2022] Available: https://scikit-learn.org/stable/modules/linear_model.html.
- [135] Scikit-learn. Naive bayes. Online, [Accessed 30/04/2022] Available: http://scikit-learn.org/stable/modules/naive_bayes.html.
- [136] Scikit-learn. Multinomialnb. Online, [Accessed 30/05/2022] Available: https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html.
- [137] Fabrizio Sebastiani. Machine learning in automated text categorization. *ACM computing surveys (CSUR)*, 34(1):1–47, 2002.
-

- [138] Priyanka Sharma and Manavjeet Kaur. Classification in pattern recognition: A review. *International Journal of Advanced Research in Computer Science and Software Engineering*, 3(4), 2013.
- [139] Sagar Sharma, Simone Sharma, and Anidhya Athaiya. Activation functions in neural networks. *towards data science*, 6(12):310–316, 2017.
- [140] Alex Shenfield, David Day, and Aladdin Ayesh. Intelligent intrusion detection systems using artificial neural networks. *Ict Express*, 4(2):95–99, 2018.
- [141] Soo-Yong Shin and Hun-Sung Kim. Data pseudonymization in a range that does not affect data quality: correlation with the degree of participation of clinicians. *Journal of Korean medical science*, 36(44), 2021.
- [142] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.
- [143] Chris J Skinner and MJ Elliot. A measure of disclosure risk for microdata. *Journal of the Royal Statistical Society: series B (statistical methodology)*, 64(4):855–867, 2002.
- [144] Marina Sokolova, Nathalie Japkowicz, and Stan Szpakowicz. Beyond accuracy, f-score and roc: a family of discriminant measures for performance evaluation. In *Australasian joint conference on artificial intelligence*, pages 1015–1021. Springer, 2006.
- [145] I. Stančín and A. Jović. An overview and comparison of free python libraries for data mining and big data analysis. In *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 977–982, 2019.
- [146] Sai Sumathi and SN Sivanandam. *Introduction to data mining and its applications*, volume 29. Springer, 2006.
- [147] Daniel Svozil, Vladimir Kvasnicka, and Jiri Pospichal. Introduction to multi-layer feed-forward neural networks. *Chemometrics and intelligent laboratory systems*, 39(1):43–62, 1997.
- [148] Latanya Sweeney. Weaving technology and policy together to maintain confidentiality. *The Journal of Law, Medicine & Ethics*, 25(2-3):98–110, 1997.
- [149] Latanya Sweeney. k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(05):557–570, 2002.
- [150] Kashvi Taunk, Sanjukta De, Srishti Verma, and Aleena Swetapadma. A brief review of nearest neighbor algorithm for learning and classification. In *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*, pages 1255–1260. IEEE, 2019.
- [151] Alaa Tharwat. Classification assessment methods. *Applied Computing and Informatics*, 2020.
- [152] Traian Marius Truta and Bindu Vinay. Privacy protection: p-sensitive k-anonymity property. In *22nd International Conference on Data Engineering Workshops (ICDEW'06)*, pages 94–94. IEEE, 2006.

-
- [153] Stephen Tu. Lecture 20: Introduction to differential privacy. *Advanced Topics in Data Processing*, pages 1–9, 2013.
- [154] Anastasia Tugaenko, Viktoria Gingina, Kira Matveeva, Mikhail Chupilko, and Sari Haj Hussein. Data anonymization techniques. 2011.
- [155] Henk CA Van Tilborg and Sushil Jajodia. *Encyclopedia of cryptography and security*. Springer Science & Business Media, 2014.
- [156] Sofia Visa, Brian Ramsay, Anca L Ralescu, and Esther Van Der Knaap. Confusion matrix-based feature selection. *MAICS*, 710:120–127, 2011.
- [157] Kuldip Vora, Shruti Yagnik, and Mtech Scholar. A survey on backpropagation algorithms for feedforward neural networks. 2014.
- [158] Sun-Chong Wang. Artificial neural network. In *Interdisciplinary computing in java programming*, pages 81–100. Springer, 2003.
- [159] Yong Wang, Julia Hodges, and Bo Tang. Classification of web documents using a naive bayes method. In *Proceedings. 15th IEEE International Conference on Tools with Artificial Intelligence*, pages 560–564. IEEE, 2003.
- [160] Professor Anita Wasilewska. Classification lecture notes, 2018.
- [161] Hilde JP Weerts, Andreas C Mueller, and Joaquin Vanschoren. Importance of tuning hyperparameters of machine learning algorithms. *arXiv preprint arXiv:2007.07588*, 2020.
- [162] Jason Wei and Kai Zou. Eda: Easy data augmentation techniques for boosting performance on text classification tasks. *arXiv preprint arXiv:1901.11196*, 2019.
- [163] RL Wilson and PA Rosen. The impact of data perturbation techniques on data mining accuracy. In *Proceedings of the 33rd Annual Meeting of the Decision Sciences Institute*, pages 181–185, 2002.
- [164] Ian H Witten, Eibe Frank, Mark A Hall, Christopher J Pal, and MINING DATA. Practical machine learning tools and techniques. In *DATA MINING*, volume 2, page 4, 2005.
- [165] Alexandra Wood, Micah Altman, Aaron Bembenek, Mark Bun, Marco Gaboardi, James Honaker, Kobbi Nissim, David R O’Brien, Thomas Steinke, and Salil Vadhan. Differential privacy: A primer for a non-technical audience. *Vand. J. Ent. & Tech. L.*, 21:209, 2018.
- [166] Shuo Xu. Bayesian naïve bayes classifiers to text classification. *Journal of Information Science*, 44(1):48–59, 2018.
- [167] Reda Yacouby and Dustin Axman. Probabilistic extension of precision, recall, and f1 score for more thorough evaluation of classification models. In *Proceedings of the First Workshop on Evaluation and Comparison of NLP Systems*, pages 79–91, 2020.
- [168] Xue Ying. An overview of overfitting and its solutions. In *Journal of Physics: Conference Series*, volume 1168, page 022022. IOP Publishing, 2019.
- [169] Xinghuo Yu, M Onder Efe, and Okyay Kaynak. A general backpropagation algorithm for feedforward neural networks learning. *IEEE transactions on neural networks*, 13(1):251–254, 2002.
-

- [170] Babak Zamanlooy and Mitra Mirhassani. Efficient vlsi implementation of neural networks with hyperbolic tangent activation function. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 22(1):39–48, 2013.
- [171] Amer Zayegh and Nizar Al Bassam. Neural network principles and applications. *Digital Systems*, 2018.
- [172] Dongxiang Zhang, Yuyang Nie, Sai Wu, Yanyan Shen, and Kian-Lee Tan. Multi-context attention for entity matching. In *Proceedings of The Web Conference 2020*, pages 2634–2640, 2020.
- [173] Harry Zhang. The optimality of naive bayes. *Aa*, 1(2):3, 2004.
- [174] Zhongheng Zhang. Introduction to machine learning: k-nearest neighbors. *Annals of translational medicine*, 4(11), 2016.
- [175] Zhiyu Zhou, Haodong Ji, and Zefei Zhu. Online sequential fuzzy dropout extreme learning machine compensate for sliding-mode control system errors of uncertain robot manipulator. *International Journal of Machine Learning and Cybernetics*, 13(8):2171–2187, 2022.
- [176] Xiaojin Jerry Zhu. Semi-supervised learning literature survey. 2005.
- [177] Xiaonan Zou, Yong Hu, Zhewen Tian, and Kaiyuan Shen. Logistic regression model optimization and case analysis. In *2019 IEEE 7th International Conference on Computer Science and Network Technology (ICCSNT)*, pages 135–139. IEEE, 2019.

Appendix A

Synthetic Dataset

A.1 Netflix Prize Data and GIC Databases

The synthetic database created for this research to emulate the Netflix Prize Data and the Governor William Weld (GIC) re-identification scenarios can be found here [74]

A.2 Data Collection Experiment Guide

Event: PhD Data Collection Experiment

Organiser: Olabayo Ishola

Research Topic: Data Mining and Re-identification: Analysis of Database Query Patterns That Pose A Threat to Anonymised Information

Supervisors: Prof Eerke Bioten & Prof Aladdin Ayesh

Participants: Members from DMU Hacker Society

The activities in this experiment will be divided into two sessions, with each session aiming to achieve different goals.

Session One

- Background information on re-identification
- Getting used to the *Netflix Prize* database: Using SQL queries, engage with the database with the aim of learning as much as you can about the database structural details

Session Two

- Using the *IMDB* database as a secondary source of information, attempt to link individual(s) in both databases (*Netflix Prize* database and *IMDB*).

Important Notes

1. According to the **Data Protection Act 2018**, performing re-identification is illegal, except for research purposes.
2. I am asking for your permission to record your SQL query data. The recorded queries will be used to support my research.
3. This experiment will record SQL query data only. No personal data is recorded. The recorded query data will not be linkable to any participant.
4. You have the option to choose whether your SQL query data is included to use for further research analysis.

References

Ishola, O., Bioten, E.A., Ayesh, A., Albakri, A. (2020) Recognising Re-identification Attacks on Databases, by Interpreting them as SQL Queries: A Technical Study. Privacy in Statistical Databases 2020 (PSD2020), Arezzo, Italy, September 2020.

Appendix B

SQL Query Data

B.1 Normal Query Sample

```
1      exec sp_reset_connection
2  go
3  SELECT
4      SERVERPROPERTY('EngineEdition') EngineEdition ,
5      SERVERPROPERTY('ProductVersion') ProductVersion
6
7  go
8  SELECT SYSTEMUSER
9  go
10 DECLARE @edition sysname;
11 SET @edition = cast(SERVERPROPERTY(N'EDITION') as sysname);
12 SELECT case when @edition = N'SQL Azure' then 2 else 1 end as '
DatabaseEngineType',
13 SERVERPROPERTY('EngineEdition') AS DatabaseEngineEdition ,
14 SERVERPROPERTY('ProductVersion') AS ProductVersion ,
15 @@MICROSOFTVERSION AS MicrosoftVersion;
16 select host_platform from sys.dm_os_host_info
17 if @edition = N'SQL Azure '
18     select 'TCP' as ConnectionProtocol
19 else
20     exec ('select CONVERT(nvarchar(40),CONNECTIONPROPERTY('net_transport')) as
ConnectionProtocol')
21
22 go
23 SET ROWCOUNT 0 SET TEXTSIZE 2147483647 SET NOCOUNT OFF SET
CONCAT_NULL_YIELDS_NULL ON SET ARITHABORT ON SET LOCK_TIMEOUT -1 SET
QUERY_GOVERNOR_COST_LIMIT 0 SET DEADLOCK_PRIORITY NORMAL SET TRANSACTION
ISOLATION_LEVEL READ COMMITTED SET ANSI_NULLS ON SET ANSI_NULL_DFLT_ON ON SET
ANSI_PADDING ON SET ANSI_WARNINGS ON SET CURSOR_CLOSE_ON_COMMIT OFF SET
IMPLICIT_TRANSACTIONS OFF SET QUOTED_IDENTIFIER ON
24 go
25 select @@spid;
26 select SERVERPROPERTY('ProductLevel');
27
28 go
29 DECLARE @edition sysname;
30 SET @edition = cast(SERVERPROPERTY(N'EDITION') as sysname);
31 SELECT case when @edition = N'SQL Azure' then 2 else 1 end as '
DatabaseEngineType',
```

```
32 SERVERPROPERTY('EngineEdition') AS DatabaseEngineEdition ,
33 SERVERPROPERTY('ProductVersion') AS ProductVersion ,
34 @@MICROSOFTVERSION AS MicrosoftVersion ;
35 select host_platform from sys.dm_os_host_info
36 if @edition = N'SQL Azure'
37     select 'TCP' as ConnectionProtocol
38 else
39     exec ('select CONVERT(nvarchar(40),CONNECTIONPROPERTY('net_transport')) as
ConnectionProtocol')
40
41 go
42 SELECT dtb.name AS [Name], dtb.state AS [State] FROM master.sys.databases dtb
43 go
44 DECLARE @edition sysname;
45 SET @edition = cast(SERVERPROPERTY(N'EDITION') as sysname);
46 SELECT case when @edition = N'SQL Azure' then 2 else 1 end as '
DatabaseEngineType',
47 SERVERPROPERTY('EngineEdition') AS DatabaseEngineEdition ,
48 SERVERPROPERTY('ProductVersion') AS ProductVersion ,
49 @@MICROSOFTVERSION AS MicrosoftVersion ;
50 select host_platform from sys.dm_os_host_info
51 if @edition = N'SQL Azure'
52     select 'TCP' as ConnectionProtocol
53 else
54     exec ('select CONVERT(nvarchar(40),CONNECTIONPROPERTY('net_transport')) as
ConnectionProtocol')
55
56 go
57 DECLARE @edition sysname;
58 SET @edition = cast(SERVERPROPERTY(N'EDITION') as sysname);
59 SELECT case when @edition = N'SQL Azure' then 2 else 1 end as '
DatabaseEngineType',
60 SERVERPROPERTY('EngineEdition') AS DatabaseEngineEdition ,
61 SERVERPROPERTY('ProductVersion') AS ProductVersion ,
62 @@MICROSOFTVERSION AS MicrosoftVersion ;
63 select host_platform from sys.dm_os_host_info
64 if @edition = N'SQL Azure'
65     select 'TCP' as ConnectionProtocol
66 else
67     exec ('select CONVERT(nvarchar(40),CONNECTIONPROPERTY('net_transport')) as
ConnectionProtocol')
68
69 go
70 select SERVERPROPERTY(N'servername')
71 go
72 use [master];
73 if (db_id() = 1)
74 begin
75     — contained auth is 0 when connected to master
76     select 0
77 end
78 else
79 begin
80     — need dynamic sql so that we compile this query only when we know resource db
is available
81     exec('select case when authenticating_database_id = 1 then 0 else 1 end from
82     sys.dm_exec_sessions where session_id = @@SPID')
83 end; use [master]; —resetting the context
84 go
```

```

85 SELECT
86 CAST(
87     serverproperty(N'Servername')
88     AS sysname) AS [Name]
89 go
90
91     declare @MasterPath nvarchar(512)
92     declare @LogPath nvarchar(512)
93     declare @ErrorLog nvarchar(512)
94     declare @ErrorLogPath nvarchar(512)
95     declare @Slash varchar = convert(varchar, serverproperty('PathSeparator
96     '))
97     if (SERVERPROPERTY('EngineEdition') = 8 /* SQL Managed Instance */)
98     begin
99         select @MasterPath=substring(physical_name, 1, len(physical_name) -
100         charindex(@Slash, reverse(physical_name))) from master.sys.database_files where
101         file_id = 1
102         select @LogPath=substring(physical_name, 1, len(physical_name) -
103         charindex(@Slash, reverse(physical_name))) from master.sys.database_files where
104         file_id = 2
105     end
106     else
107     begin
108         select @MasterPath=substring(physical_name, 1, len(physical_name) -
109         charindex(@Slash, reverse(physical_name))) from master.sys.database_files where
110         name=N'master'
111         select @LogPath=substring(physical_name, 1, len(physical_name) -
112         charindex(@Slash, reverse(physical_name))) from master.sys.database_files where
113         name=N'mastlog'
114     end
115     select @ErrorLog=cast(SERVERPROPERTY(N'errorlogfilename') as nvarchar
116     (512))
117     select @ErrorLogPath=IIF(@ErrorLog IS NULL, N'', substring(@ErrorLog,
118     1, len(@ErrorLog) - charindex(@Slash, reverse(@ErrorLog))))
119
120
121     declare @SmoRoot nvarchar(512)
122     exec master.dbo.xp_instance_regread N'HKEY_LOCAL_MACHINE', N'SOFTWARE\
123     Microsoft\MSSQLServer\Setup', N'SQLPath', @SmoRoot OUTPUT
124
125
126 SELECT
127 CAST(case when 'a' <> 'A' then 1 else 0 end AS bit) AS [IsCaseSensitive],
128 @@MAX_PRECISION AS [MaxPrecision],
129 @ErrorLogPath AS [ErrorLogPath],
130 @SmoRoot AS [RootDirectory],
131 SERVERPROPERTY('PathSeparator') AS [PathSeparator],
132 CAST(FULLTEXTSERVICEPROPERTY('IsFullTextInstalled') AS bit) AS [
133 IsFullTextInstalled],
134 @LogPath AS [MasterDBLogPath],
135 @MasterPath AS [MasterDBPath],
136 SERVERPROPERTY(N'ProductVersion') AS [VersionString],
137 CAST(SERVERPROPERTY(N'Edition') AS sysname) AS [Edition],
138 CAST(SERVERPROPERTY(N'ProductLevel') AS sysname) AS [ProductLevel],
139 CAST(SERVERPROPERTY(N'IsSingleUser') AS bit) AS [IsSingleUser],
140 CAST(SERVERPROPERTY('EngineEdition') AS int) AS [EngineEdition],
141 convert(sysname, serverproperty(N'collation')) AS [Collation],

```

```

131 CAST(ISNULL(SERVERPROPERTY(N'MachineName'),N'') AS sysname) AS [NetName],
132 CAST(ISNULL(SERVERPROPERTY('IsClustered'),N'') AS bit) AS [IsClustered],
133 SERVERPROPERTY(N'ResourceVersion') AS [ResourceVersionString],
134 SERVERPROPERTY(N'ResourceLastUpdateDateTime') AS [ResourceLastUpdateDateTime],
135 SERVERPROPERTY(N'CollationID') AS [CollationID],
136 SERVERPROPERTY(N'ComparisonStyle') AS [ComparisonStyle],
137 SERVERPROPERTY(N'SqlCharSet') AS [SqlCharSet],
138 SERVERPROPERTY(N'SqlCharSetName') AS [SqlCharSetName],
139 SERVERPROPERTY(N'SqlSortOrder') AS [SqlSortOrder],
140 SERVERPROPERTY(N'SqlSortOrderName') AS [SqlSortOrderName],
141 SERVERPROPERTY(N'BuildClrVersion') AS [BuildClrVersionString],
142 ISNULL(SERVERPROPERTY(N'ComputerNamePhysicalNetBIOS'),N'') AS [
ComputerNamePhysicalNetBIOS],
143 CAST(SERVERPROPERTY('IsPolyBaseInstalled') AS bit) AS [IsPolyBaseInstalled]
144 go
145 SELECT
146 dtb.name AS [Name],
147 dtb.database_id AS [ID],
148 CAST(case when dtb.name in ('master','model','msdb','tempdb') then 1 else dtb.
is_distributor end AS bit) AS [IsSystemObject],
149 dtb.collation_name AS [Collation],
150 CAST(has_dbaccess(dtb.name) AS bit) AS [IsAccessible],
151 dtb.name AS [DatabaseName2]
152 FROM
153 master.sys.databases AS dtb
154 ORDER BY
155 [Name] ASC
156 go
157 SELECT
158 dtb.name AS [Name],
159 dtb.database_id AS [ID],
160 CAST(has_dbaccess(dtb.name) AS bit) AS [IsAccessible]
161 FROM
162 master.sys.databases AS dtb
163 ORDER BY
164 [Name] ASC
165 go
166 SELECT
167 log.name AS [Name],
168 log.principal_id AS [ID],
169 ISNULL(ak.name,N'') AS [AsymmetricKey],
170 ISNULL(cert.name,N'') AS [Certificate],
171 ISNULL(c.name,N'') AS [Credential],
172 CASE WHEN N'U' = log.type THEN 0 WHEN N'G' = log.type THEN 1 WHEN N'S' = log.
type THEN 2 WHEN N'C' = log.type THEN 3 WHEN N'K' = log.type THEN 4 WHEN N'E'
= log.type THEN 5 WHEN N'X' = log.type THEN 6 END AS [LoginType]
173 FROM
174 sys.server_principals AS log
175 LEFT OUTER JOIN master.sys.asymmetric_keys AS ak ON ak.sid = log.sid
176 LEFT OUTER JOIN master.sys.certificates AS cert ON cert.sid = log.sid
177 LEFT OUTER JOIN sys.credentials AS c ON c.credential_id = log.credential_id
178 WHERE
179 (log.type in ('U', 'G', 'S', 'C', 'K', 'E', 'X') AND log.principal_id not
between 101 and 255 AND log.name <> N'##MS-AgentSigningCertificate##')
180 ORDER BY
181 [Name] ASC
182 go
183 SELECT
184 tr.name AS [Name],

```

```

185 tr.object_id AS [ID],
186 CAST(
187     tr.is_ms_shipped
188     AS bit) AS [IsSystemObject],
189 CASE WHEN tr.type = N'TR' THEN 1 WHEN tr.type = N'TA' THEN 2 ELSE 1 END AS [
ImplementationType],
190 CAST(CASE WHEN ssmod.definition IS NULL THEN 1 ELSE 0 END AS bit) AS [
IsEncrypted]
191 FROM
192 master.sys.server_triggers AS tr
193 LEFT OUTER JOIN master.sys.server_assembly_modules AS mod ON mod.object_id = tr
.object_id
194 LEFT OUTER JOIN sys.server_sql_modules AS ssmod ON ssmod.object_id = tr.
object_id
195 WHERE
196 (tr.parent_class = 100)
197 ORDER BY
198 [Name] ASC
199 go
200 use [Netflix Prize Data ALPHA]
201 go
202 SELECT
203 rl.name AS [Name]
204 FROM
205 sys.database_principals AS rl
206 WHERE
207 (rl.type = 'A')
208 ORDER BY
209 [Name] ASC
210 go
211 SELECT
212 rl.name AS [Name]
213 FROM
214 sys.database_principals AS rl
215 WHERE
216 (rl.type = 'R')
217 ORDER BY
218 [Name] ASC
219 go
220 SELECT
221 (@@microsoftversion / 0x1000000) & 0xff AS [VersionMajor]
222 go
223 SELECT
224 s.name AS [Name]
225 FROM
226 sys.schemas AS s
227 ORDER BY
228 [Name] ASC
229 go
230 SELECT
231 tr.name AS [Name],
232 tr.object_id AS [ID],
233 CAST(
234     tr.is_ms_shipped
235     AS bit) AS [IsSystemObject],
236 CASE WHEN tr.type = N'TR' THEN 1 WHEN tr.type = N'TA' THEN 2 ELSE 1 END AS [
ImplementationType],
237 CAST(CASE WHEN ISNULL(smtr.definition , ssmtr.definition) IS NULL THEN 1 ELSE 0
END AS bit) AS [IsEncrypted]

```

```
238 FROM
239 sys.triggers AS tr
240 LEFT OUTER JOIN sys.sql_modules AS smtr ON smtr.object_id = tr.object_id
241 LEFT OUTER JOIN sys.system_sql_modules AS ssmtr ON ssmtr.object_id = tr.
object_id
242 WHERE
243 (tr.parent_class = 0)
244 ORDER BY
245 [Name] ASC
246 go
247 SELECT
248 u.name AS [Name],
249 u.principal_id AS [ID],
250 ISNULL(ak.name,N'') AS [AsymmetricKey],
251 ISNULL(cert.name,N'') AS [Certificate],
252 ISNULL(suser_sname(u.sid),N'') AS [Login],
253
254         CASE
255         WHEN N'C' = u.type THEN 1
256         WHEN N'K' = u.type THEN 2
257         WHEN N'S' = u.type AND SUSER_SNAME(u.sid) is null AND u.
authentication_type != 2 THEN 3
258         ELSE 0 END
259         AS [UserType]
260 FROM
261 sys.database_principals AS u
262 LEFT OUTER JOIN sys.asymmetric_keys AS ak ON ak.sid = u.sid
263 LEFT OUTER JOIN sys.certificates AS cert ON cert.sid = u.sid
264 WHERE
265 (u.type in ('U', 'S', 'G', 'C', 'K', 'E', 'X'))
266 ORDER BY
267 [Name] ASC
268 go
269 SELECT CASE WHEN has_dbaccess(N'Netflix Prize Data ALPHA') = 1 THEN 'true' ELSE
'false' END
270 go
271 SELECT
272 (select schema_name()) AS [DefaultSchema]
273 go
274 SELECT
275 SCHEMA_NAME(tbl.schema_id) AS [Schema],
276 tbl.name AS [Name],
277 tbl.object_id AS [ID]
278 FROM
279 sys.tables AS tbl
280 ORDER BY
281 [Schema] ASC,[Name] ASC
282 go
283 SELECT
284 SCHEMA_NAME(obj.schema_id) AS [Schema],
285 obj.name AS [Name],
286 obj.object_id AS [ID],
287 usrt.name AS [DataType],
288 ISNULL(baset.name, N'') AS [SystemType],
289 CAST(CASE WHEN baset.name IN (N'nchar', N'nvarchar') AND ret_param.max_length
<> -1 THEN ret_param.max_length/2 ELSE ret_param.max_length END AS int) AS [
Length],
290 CAST(ret_param.precision AS int) AS [NumericPrecision],
291 CAST(ret_param.scale AS int) AS [NumericScale],
```



```

292 ISNULL(xcret_param.name, N'') AS [XmlSchemaNamespace],
293 ISNULL(s2ret_param.name, N'') AS [XmlSchemaNamespaceSchema],
294 ISNULL( (case ret_param.is_xml_document when 1 then 2 else 1 end), 0) AS [
  XmlDocumentConstraint],
295 slret_param.name AS [DataTypeSchema]
296 FROM
297 sys.objects AS obj
298 LEFT OUTER JOIN sys.all_parameters AS ret_param ON ret_param.object_id = obj.
  object_id and ret_param.is_output = 1
299 LEFT OUTER JOIN sys.types AS usrt ON usrt.user_type_id = ret_param.user_type_id
300 LEFT OUTER JOIN sys.types AS baset ON (baset.user_type_id = ret_param.
  system_type_id and baset.user_type_id = baset.system_type_id) or ((baset.
  system_type_id = ret_param.system_type_id) and (baset.user_type_id = ret_param.
  user_type_id) and (baset.is_user_defined = 0) and (baset.is_assembly_type = 1))
301 LEFT OUTER JOIN sys.xml_schema_collections AS xcret_param ON xcret_param.
  xml_collection_id = ret_param.xml_collection_id
302 LEFT OUTER JOIN sys.schemas AS s2ret_param ON s2ret_param.schema_id =
  xcret_param.schema_id
303 LEFT OUTER JOIN sys.schemas AS slret_param ON slret_param.schema_id = usrt.
  schema_id
304 WHERE
305 (obj.type=N'AF')
306 ORDER BY
307 [Schema] ASC,[Name] ASC
308 go
309 SELECT
310 SCHEMA_NAME(s.schema_id) AS [Schema],
311 s.name AS [Name],
312 s.object_id AS [ID],
313 N'' AS [BaseDatabase],
314 N'' AS [BaseObject],
315 N'' AS [BaseSchema],
316 N'' AS [BaseServer],
317
318             CASE OBJECTPROPERTYEX(s.object_id, 'BaseType')
319                 WHEN N'U' THEN 1
320                 WHEN N'V' THEN 2
321                 WHEN N'P' THEN 3
322                 WHEN N'FN' THEN 4
323                 WHEN N'TF' THEN 5
324                 WHEN N'IF' THEN 6
325                 WHEN N'X' THEN 7
326                 WHEN N'RF' THEN 8
327                 WHEN N'PC' THEN 9
328                 WHEN N'FS' THEN 10
329                 WHEN N'FT' THEN 11
330                 WHEN N'AF' THEN 12 ELSE 0 END
331             AS [BaseType],
332 s.base_object_name AS [BaseObjectName]
333 FROM
334 sys.synonyms AS s
335 ORDER BY
336 [Schema] ASC,[Name] ASC
337 go
338 SELECT
339 SCHEMA_NAME(xproc.schema_id) AS [Schema],
340 xproc.name AS [Name],
341 xproc.object_id AS [ID],
342 CAST(

```

```
343         xproc.is_ms_shipped
344         AS bit) AS [IsSystemObject]
345 FROM
346 sys.all_objects AS xproc
347 WHERE
348 (xproc.type='X')
349 ORDER BY
350 [Schema] ASC,[Name] ASC
351 go
352 SELECT
353 sst.name AS [Schema],
354 st.name AS [Name]
355 FROM
356 sys.types AS st
357 INNER JOIN sys.schemas AS sst ON sst.schema_id = st.schema_id
358 WHERE
359 (st.schema_id!=4 and st.system_type_id!=240 and st.user_type_id != st.
360 system_type_id and st.is_table_type != 1)
361 ORDER BY
362 [Schema] ASC,[Name] ASC
363 go
364 SELECT
365 SCHEMA_NAME(tt.schema_id) AS [Schema],
366 tt.name AS [Name]
367 FROM
368 sys.table_types AS tt
369 INNER JOIN sys.schemas AS stt ON stt.schema_id = tt.schema_id
370 ORDER BY
371 [Schema] ASC,[Name] ASC
372 go
373 SELECT
374 satypes.name AS [Schema],
375 atypes.name AS [Name]
376 FROM
377 sys.assembly_types AS atypes
378 INNER JOIN sys.assemblies AS asmb1 ON (asmb1.assembly_id = atypes.assembly_id)
379 and (atypes.is_user_defined = 1)
380 INNER JOIN sys.schemas AS satypes ON satypes.schema_id = atypes.schema_id
381 ORDER BY
382 [Schema] ASC,[Name] ASC
383 go
384 use [Netflix Prize Data ALPHA]
385 go
386 SELECT
387     TABLE_NAME
388 FROM
389     [Netflix Prize Data ALPHA].TABLES
390 go
391 SELECT @@SPID;
392 go
393 SELECT
394     *
395 FROM
396 INFORMATION_SCHEMA.TABLES
397 go
398 use [Netflix Prize Data ALPHA]
399 go
```

```

400 SELECT @@SPID;
401 go
402 SELECT
403     TABLE_NAME,
404     COLUMN_NAME
405 FROM
406     INFORMATION_SCHEMA.COLUMNS
407 go
408 SELECT @@SPID;
409 go
410 SELECT
411     *
412 FROM
413 INFORMATION_SCHEMA.TABLES;
414 go
415 SELECT
416     dtb.name AS [Name],
417     dtb.database_id AS [ID],
418     CAST(has_dbaccess(dtb.name) AS bit) AS [IsAccessible]
419 FROM
420     master.sys.databases AS dtb
421 ORDER BY
422     [Name] ASC
423 go
424 use [Netflix Prize Data ALPHA]
425 go
426 SET DEADLOCK_PRIORITY -10
427 go
428 SELECT target_data
429
430 dm_xe_session_targets xet WITH(nolock)
431
432 dm_xe_sessions xes WITH(nolock)
433
434 address = xet.event_session_address
435
436 name = 'telemetry_xevents'
437
438 target_name = 'ring_buffer'
439
440 go
441 SET DEADLOCK_PRIORITY -10
442 go
443 if not exists (select * from sys.dm_xe_sessions where name = 'telemetry_xevents'
444 ')
445     alter event session telemetry_xevents on server state=start
446 go
447 SELECT @@SPID;
448 go
449 select schema_name(t.schema_id) as schema_name,
450     t.name as table_name,
451     t.create_date,
452     t.modify_date
453 from sys.tables t
454 where schema_name(t.schema_id) = 'Netflix Prize Data ALPHA' — put schema name
455 here
456 order by table_name;
457 go
458 SELECT @@SPID;
459 go

```

```
452 select schema_name(t.schema_id) as schema_name ,
453         t.name as table_name ,
454         t.create_date ,
455         t.modify_date
456 from sys.tables t
457 where table_name = 'Netflix Prize Data ALPHA' — put schema name here
458 order by table_name;
459 go
460 SELECT
461 dtb.name AS [Name] ,
462 dtb.database_id AS [ID] ,
463 CAST(has_dbaccess(dtb.name) AS bit) AS [IsAccessible]
464 FROM
465 master.sys.databases AS dtb
466 ORDER BY
467 [Name] ASC
468 go
469 SELECT @@SPID;
470 go
471 select schema_name(t.schema_id) as schema_name ,
472         t.name as table_name ,
473         t.create_date ,
474         t.modify_date
475 from sys.tables t
476 —where t.table_name = 'Netflix Prize Data ALPHA' — put schema name here
477 order by table_name;
478 go
479 SELECT
480 dtb.name AS [Name] ,
481 dtb.database_id AS [ID] ,
482 CAST(has_dbaccess(dtb.name) AS bit) AS [IsAccessible]
483 FROM
484 master.sys.databases AS dtb
485 ORDER BY
486 [Name] ASC
487 go
488 SELECT @@SPID;
489 go
490
491 select *
492 from sys.tables t
493 order by table_name;
494 go
495 SELECT @@SPID;
496 go
497
498 select *
499 from sys.tables ;
500 go
501 use [Netflix Prize Data ALPHA]
502 go
503 SELECT @@SPID;
504 go
505
506     SELECT
507     *
508 FROM
509     SYSOBJECTS
510 WHERE
```

```

511     xtype = 'U';
512
513 go
514 SELECT @@SPID;
515 go
516
517 SELECT
518     name,
519     crdate
520 FROM
521     SYSOBJECTS
522 WHERE
523     xtype = 'U';
524
525 go
526 SELECT
527     dtb.name AS [Name],
528     dtb.database_id AS [ID],
529     CAST(has_dbaccess(dtb.name) AS bit) AS [IsAccessible]
530 FROM
531     master.sys.databases AS dtb
532 ORDER BY
533     [Name] ASC
534 go
535 SELECT @@SPID;
536 go
537
538 SELECT
539     *
540 FROM
541     INFORMATION_SCHEMA.TABLES;
542
543 go
544 SET DEADLOCK_PRIORITY -10
545 go
546 SELECT target_data
547
548 dm_xe_session_targets xet WITH(nolock)
549
550 address = xet.event_session_address
551
552 name = 'telemetry_xevents'
553
554 target_name = 'ring_buffer'
555
556 dm_xe_sessions xes WITH(nolock)
557
558 name = 'telemetry_xevents'
559
560 FROM sys.
561 JOIN sys.
562 ON xes.
563 WHERE xes.
564 AND xet.
565
566 alter event session telemetry_xevents on server state=start
567 go
568 SET LOCK_TIMEOUT 10000
569 go
570 DECLARE @edition sysname;
571 SET @edition = cast(SERVERPROPERTY(N'EDITION') as sysname);
572 SELECT case when @edition = N'SQL Azure' then 2 else 1 end as '
DatabaseEngineType',

```

```
563 SERVERPROPERTY('EngineEdition') AS DatabaseEngineEdition ,
564 SERVERPROPERTY('ProductVersion') AS ProductVersion ,
565 @@MICROSOFTVERSION AS MicrosoftVersion ;
566 select host_platform from sys.dm_os_host_info
567 if @edition = N'SQL Azure'
568     select 'TCP' as ConnectionProtocol
569 else
570     exec ('select CONVERT(nvarchar(40),CONNECTIONPROPERTY('net_transport')) as
ConnectionProtocol')
571
572 go
573 IF OBJECT_ID (N'[sys].[database_query_store_options]') IS NOT NULL BEGIN SELECT
ISNULL(actual_state , -2) FROM sys.database_query_store_options; IF EXISTS (
SELECT TOP(1) 1 FROM sys.query_store_runtime_stats) SELECT 1 ELSE SELECT 0; END
574 go
575 SET LOCK_TIMEOUT 10000
576 go
577 DECLARE @edition sysname;
578 SET @edition = cast(SERVERPROPERTY(N'EDITION') as sysname);
579 SELECT case when @edition = N'SQL Azure' then 2 else 1 end as '
DatabaseEngineType' ,
580 SERVERPROPERTY('EngineEdition') AS DatabaseEngineEdition ,
581 SERVERPROPERTY('ProductVersion') AS ProductVersion ,
582 @@MICROSOFTVERSION AS MicrosoftVersion ;
583 select host_platform from sys.dm_os_host_info
584 if @edition = N'SQL Azure'
585     select 'TCP' as ConnectionProtocol
586 else
587     exec ('select CONVERT(nvarchar(40),CONNECTIONPROPERTY('net_transport')) as
ConnectionProtocol')
588
589 go
590 use [Netflix Prize Data ALPHA]
591 go
592 SET LOCK_TIMEOUT 10000
593 go
594 DECLARE @edition sysname;
595 SET @edition = cast(SERVERPROPERTY(N'EDITION') as sysname);
596 SELECT case when @edition = N'SQL Azure' then 2 else 1 end as '
DatabaseEngineType' ,
597 SERVERPROPERTY('EngineEdition') AS DatabaseEngineEdition ,
598 SERVERPROPERTY('ProductVersion') AS ProductVersion ,
599 @@MICROSOFTVERSION AS MicrosoftVersion ;
600 select host_platform from sys.dm_os_host_info
601 if @edition = N'SQL Azure'
602     select 'TCP' as ConnectionProtocol
603 else
604     exec ('select CONVERT(nvarchar(40),CONNECTIONPROPERTY('net_transport')) as
ConnectionProtocol')
605
606 go
607 use [Netflix Prize Data ALPHA]
608 go
609 SET LOCK_TIMEOUT 10000
610 go
611 DECLARE @edition sysname;
612 SET @edition = cast(SERVERPROPERTY(N'EDITION') as sysname);
613 SELECT case when @edition = N'SQL Azure' then 2 else 1 end as '
DatabaseEngineType' ,
```

```

614 SERVERPROPERTY('EngineEdition') AS DatabaseEngineEdition ,
615 SERVERPROPERTY('ProductVersion') AS ProductVersion ,
616 @@MICROSOFTVERSION AS MicrosoftVersion ;
617 select host_platform from sys.dm_os_host_info
618 if @edition = N'SQL Azure'
619     select 'TCP' as ConnectionProtocol
620 else
621     exec ('select CONVERT(nvarchar(40),CONNECTIONPROPERTY('net_transport')) as
ConnectionProtocol')
622
623 go
624 use [Netflix Prize Data ALPHA]
625 go
626 SET LOCK_TIMEOUT 10000
627 go
628 DECLARE @edition sysname;
629 SET @edition = cast(SERVERPROPERTY(N'EDITION') as sysname);
630 SELECT case when @edition = N'SQL Azure' then 2 else 1 end as '
DatabaseEngineType' ,
631 SERVERPROPERTY('EngineEdition') AS DatabaseEngineEdition ,
632 SERVERPROPERTY('ProductVersion') AS ProductVersion ,
633 @@MICROSOFTVERSION AS MicrosoftVersion ;
634 select host_platform from sys.dm_os_host_info
635 if @edition = N'SQL Azure'
636     select 'TCP' as ConnectionProtocol
637 else
638     exec ('select CONVERT(nvarchar(40),CONNECTIONPROPERTY('net_transport')) as
ConnectionProtocol')
639
640 go
641 use [Netflix Prize Data ALPHA]
642 go
643 DECLARE @edition sysname;
644 SET @edition = cast(SERVERPROPERTY(N'EDITION') as sysname);
645 SELECT case when @edition = N'SQL Azure' then 2 else 1 end as '
DatabaseEngineType' ,
646 SERVERPROPERTY('EngineEdition') AS DatabaseEngineEdition ,
647 SERVERPROPERTY('ProductVersion') AS ProductVersion ,
648 @@MICROSOFTVERSION AS MicrosoftVersion ;
649 select host_platform from sys.dm_os_host_info
650 if @edition = N'SQL Azure'
651     select 'TCP' as ConnectionProtocol
652 else
653     exec ('select CONVERT(nvarchar(40),CONNECTIONPROPERTY('net_transport')) as
ConnectionProtocol')
654
655 go
656 use [Netflix Prize Data ALPHA]
657 go
658 SELECT @@SPID;
659 go
660 select * from [dbo].[MovieTitles$];
661 go
662 use [Netflix Prize Data ALPHA]
663 go
664 SELECT @@SPID;
665 go
666 select * from [dbo].[TrainingData$];
667 go

```

```

668 SELECT @@SPID;
669 go
670 select * from [dbo].[MovieTitles$];
671 go
672 SELECT @@SPID;
673 go
674 select * from [dbo].[TrainingData$];
675 go
676 SELECT
677 dtb.name AS [Name],
678 dtb.database_id AS [ID],
679 CAST(has_dbaccess(dtb.name) AS bit) AS [IsAccessible]
680 FROM
681 master.sys.databases AS dtb
682 ORDER BY
683 [Name] ASC
684 go
685 SELECT @@SPID;
686 go
687 select count (1) from [dbo].[MovieTitles$];
688 go
689 SELECT @@SPID;
690 go
691 select count(1) from [dbo].[TrainingData$]; --
692 go

```

B.2 Re-identification Attempt Query Sample

```

1 SET LOCK_TIMEOUT 10000
2 go
3 DECLARE @edition sysname;
4 SET @edition = cast(SERVERPROPERTY(N'EDITION') as sysname);
5 SELECT case when @edition = N'SQL Azure' then 2 else 1 end as '
   DatabaseEngineType',
6 SERVERPROPERTY('EngineEdition') AS DatabaseEngineEdition,
7 SERVERPROPERTY('ProductVersion') AS ProductVersion,
8 @@MICROSOFTVERSION AS MicrosoftVersion;
9 select host_platform from sys.dm_os_host_info
10 if @edition = N'SQL Azure'
11   select 'TCP' as ConnectionProtocol
12 else
13   exec ('select CONVERT(nvarchar(40),CONNECTIONPROPERTY(''net_transport'')) as
   ConnectionProtocol')
14
15 go
16 IF OBJECT_ID (N'[sys].[database_query_store_options]') IS NOT NULL BEGIN SELECT
   ISNULL(actual_state, -2) FROM sys.database_query_store_options; IF EXISTS (
   SELECT TOP(1) 1 FROM sys.query_store_runtime_stats) SELECT 1 ELSE SELECT 0; END
17 go
18 SELECT
19 dtb.name AS [Name],
20 dtb.database_id AS [ID],
21 CAST(has_dbaccess(dtb.name) AS bit) AS [IsAccessible]
22 FROM
23 master.sys.databases AS dtb
24 ORDER BY

```



```

25 [Name] ASC
26 go
27 SET LOCK_TIMEOUT 10000
28 go
29 DECLARE @edition sysname;
30 SET @edition = cast(SERVERPROPERTY(N'EDITION') as sysname);
31 SELECT case when @edition = N'SQL Azure' then 2 else 1 end as '
DatabaseEngineType',
32 SERVERPROPERTY('EngineEdition') AS DatabaseEngineEdition ,
33 SERVERPROPERTY('ProductVersion') AS ProductVersion ,
34 @@MICROSOFTVERSION AS MicrosoftVersion;
35 select host_platform from sys.dm_os_host_info
36 if @edition = N'SQL Azure'
37     select 'TCP' as ConnectionProtocol
38 else
39     exec ('select CONVERT(nvarchar(40),CONNECTIONPROPERTY('net_transport')) as
ConnectionProtocol')
40
41 go
42 use [Netflix Prize Data BETA]
43 go
44 SET LOCK_TIMEOUT 10000
45 go
46 DECLARE @edition sysname;
47 SET @edition = cast(SERVERPROPERTY(N'EDITION') as sysname);
48 SELECT case when @edition = N'SQL Azure' then 2 else 1 end as '
DatabaseEngineType',
49 SERVERPROPERTY('EngineEdition') AS DatabaseEngineEdition ,
50 SERVERPROPERTY('ProductVersion') AS ProductVersion ,
51 @@MICROSOFTVERSION AS MicrosoftVersion;
52 select host_platform from sys.dm_os_host_info
53 if @edition = N'SQL Azure'
54     select 'TCP' as ConnectionProtocol
55 else
56     exec ('select CONVERT(nvarchar(40),CONNECTIONPROPERTY('net_transport')) as
ConnectionProtocol')
57
58 go
59 use [Netflix Prize Data BETA]
60 go
61 SET LOCK_TIMEOUT 10000
62 go
63 DECLARE @edition sysname;
64 SET @edition = cast(SERVERPROPERTY(N'EDITION') as sysname);
65 SELECT case when @edition = N'SQL Azure' then 2 else 1 end as '
DatabaseEngineType',
66 SERVERPROPERTY('EngineEdition') AS DatabaseEngineEdition ,
67 SERVERPROPERTY('ProductVersion') AS ProductVersion ,
68 @@MICROSOFTVERSION AS MicrosoftVersion;
69 select host_platform from sys.dm_os_host_info
70 if @edition = N'SQL Azure'
71     select 'TCP' as ConnectionProtocol
72 else
73     exec ('select CONVERT(nvarchar(40),CONNECTIONPROPERTY('net_transport')) as
ConnectionProtocol')
74
75 go
76 use [Netflix Prize Data BETA]
77 go

```

```
78 SET LOCK_TIMEOUT 10000
79 go
80 DECLARE @edition sysname;
81 SET @edition = cast(SERVERPROPERTY(N'EDITION') as sysname);
82 SELECT case when @edition = N'SQL Azure' then 2 else 1 end as '
DatabaseEngineType',
83 SERVERPROPERTY('EngineEdition') AS DatabaseEngineEdition,
84 SERVERPROPERTY('ProductVersion') AS ProductVersion,
85 @@MICROSOFTVERSION AS MicrosoftVersion;
86 select host_platform from sys.dm_os_host_info
87 if @edition = N'SQL Azure'
88     select 'TCP' as ConnectionProtocol
89 else
90     exec ('select CONVERT(nvarchar(40),CONNECTIONPROPERTY('net_transport')) as
ConnectionProtocol')
91
92 go
93 use [Netflix Prize Data BETA]
94 go
95 DECLARE @edition sysname;
96 SET @edition = cast(SERVERPROPERTY(N'EDITION') as sysname);
97 SELECT case when @edition = N'SQL Azure' then 2 else 1 end as '
DatabaseEngineType',
98 SERVERPROPERTY('EngineEdition') AS DatabaseEngineEdition,
99 SERVERPROPERTY('ProductVersion') AS ProductVersion,
100 @@MICROSOFTVERSION AS MicrosoftVersion;
101 select host_platform from sys.dm_os_host_info
102 if @edition = N'SQL Azure'
103     select 'TCP' as ConnectionProtocol
104 else
105     exec ('select CONVERT(nvarchar(40),CONNECTIONPROPERTY('net_transport')) as
ConnectionProtocol')
106
107 go
108 SELECT @@SPID;
109 go
110 select * from [dbo].[IMDb$];
111 go
112 SELECT dtb.name AS [Name], dtb.state AS [State] FROM master.sys.databases dtb
113 go
114 SELECT dtb.name AS [Name], dtb.state AS [State] FROM master.sys.databases dtb
115 go
116 SELECT dtb.name AS [Name], dtb.state AS [State] FROM master.sys.databases dtb
117 go
118 SELECT dtb.name AS [Name], dtb.state AS [State] FROM master.sys.databases dtb
119 go
120 SET NOEXEC, PARSEONLY, FMTONLY OFF
121 go
122 SET SHOWPLAN_TEXT OFF
123 go
124 SET SHOWPLAN_ALL OFF
125 go
126 SET SHOWPLAN_XML OFF
127 go
128 use [Netflix Prize Data BETA]
129 go
130 SELECT dtb.name AS [Name], dtb.state AS [State] FROM master.sys.databases dtb
131 go
132 use [Netflix Prize Data BETA]
```

```
133 go
134 SELECT
135   rl.name AS [Name]
136 FROM
137   sys.database_principals AS rl
138 WHERE
139   (rl.type = 'A')
140 ORDER BY
141   [Name] ASC
142 go
143 use [Netflix Prize Data BETA]
144 go
145 SELECT
146   rl.name AS [Name]
147 FROM
148   sys.database_principals AS rl
149 WHERE
150   (rl.type = 'R')
151 ORDER BY
152   [Name] ASC
153 go
154 use [Netflix Prize Data BETA]
155 go
156 SELECT
157   s.name AS [Name]
158 FROM
159   sys.schemas AS s
160 ORDER BY
161   [Name] ASC
162 go
163 use [Netflix Prize Data BETA]
164 go
165 SELECT
166   tr.name AS [Name],
167   tr.object_id AS [ID],
168   CAST(
169     tr.is_ms_shipped
170     AS bit) AS [IsSystemObject],
171   CASE WHEN tr.type = N'TR' THEN 1 WHEN tr.type = N'TA' THEN 2 ELSE 1 END AS [
172     ImplementationType],
173   CAST(CASE WHEN ISNULL(smtr.definition , ssmtr.definition) IS NULL THEN 1 ELSE 0
174     END AS bit) AS [IsEncrypted]
175 FROM
176   sys.triggers AS tr
177   LEFT OUTER JOIN sys.sql_modules AS smtr ON smtr.object_id = tr.object_id
178   LEFT OUTER JOIN sys.system_sql_modules AS ssmtr ON ssmtr.object_id = tr.
179   object_id
180 WHERE
181   (tr.parent_class = 0)
182 ORDER BY
183   [Name] ASC
184 go
185 use [Netflix Prize Data BETA]
186 go
187 SELECT
188   u.name AS [Name],
189   u.principal_id AS [ID],
190   ISNULL(ak.name,N'') AS [AsymmetricKey],
191   ISNULL(cert.name,N'') AS [Certificate],
```

```

189 ISNULL(suser_sname(u.sid),N'') AS [Login],
190
191     CASE
192     WHEN N'C' = u.type THEN 1
193     WHEN N'K' = u.type THEN 2
194     WHEN N'S' = u.type AND SUSER_SNAME(u.sid) is null AND u.
authentication_type != 2 THEN 3
195     ELSE 0 END
196     AS [UserType]
197 FROM
198 sys.database_principals AS u
199 LEFT OUTER JOIN sys.asymmetric_keys AS ak ON ak.sid = u.sid
200 LEFT OUTER JOIN sys.certificates AS cert ON cert.sid = u.sid
201 WHERE
202 (u.type in ('U', 'S', 'G', 'C', 'K', 'E', 'X'))
203 ORDER BY
204 [Name] ASC
205 go
206 use [Netflix Prize Data BETA]
207 go
208 SELECT CASE WHEN has_dbaccess(N'Netflix Prize Data BETA') = 1 THEN 'true' ELSE
'false' END
209 go
210 use [Netflix Prize Data BETA];
211 go
212 SELECT
213 (select schema_name()) AS [DefaultSchema]
214 go
215 use [master];
216 go
217 use [Netflix Prize Data BETA]
218 go
219 SELECT
220 SCHEMA_NAME(tbl.schema_id) AS [Schema],
221 tbl.name AS [Name],
222 tbl.object_id AS [ID]
223 FROM
224 sys.tables AS tbl
225 ORDER BY
226 [Schema] ASC,[Name] ASC
227 go
228 SELECT
229 SCHEMA_NAME(obj.schema_id) AS [Schema],
230 obj.name AS [Name],
231 obj.object_id AS [ID],
232 usrt.name AS [DataType],
233 ISNULL(baset.name, N'') AS [SystemType],
234 CAST(CASE WHEN baset.name IN (N'nchar', N'nvarchar') AND ret_param.max_length
< -1 THEN ret_param.max_length/2 ELSE ret_param.max_length END AS int) AS [
Length],
235 CAST(ret_param.precision AS int) AS [NumericPrecision],
236 CAST(ret_param.scale AS int) AS [NumericScale],
237 ISNULL(xscret_param.name, N'') AS [XmlSchemaNamespace],
238 ISNULL(s2ret_param.name, N'') AS [XmlSchemaNamespaceSchema],
239 ISNULL( (case ret_param.is_xml_document when 1 then 2 else 1 end), 0) AS [
XmlDocumentConstraint],
240 s1ret_param.name AS [DataTypeSchema]
241 FROM
242 sys.objects AS obj

```

```

243 LEFT OUTER JOIN sys.all_parameters AS ret_param ON ret_param.object_id = obj.
object_id and ret_param.is_output = 1
244 LEFT OUTER JOIN sys.types AS usrt ON usrt.user_type_id = ret_param.user_type_id
245 LEFT OUTER JOIN sys.types AS baset ON (baset.user_type_id = ret_param.
system_type_id and baset.user_type_id = baset.system_type_id) or ((baset.
system_type_id = ret_param.system_type_id) and (baset.user_type_id = ret_param.
user_type_id) and (baset.is_user_defined = 0) and (baset.is_assembly_type = 1))
246 LEFT OUTER JOIN sys.xml_schema_collections AS xscret_param ON xscret_param.
xml_collection_id = ret_param.xml_collection_id
247 LEFT OUTER JOIN sys.schemas AS s2ret_param ON s2ret_param.schema_id =
xscret_param.schema_id
248 LEFT OUTER JOIN sys.schemas AS slret_param ON slret_param.schema_id = usrt.
schema_id
249 WHERE
250 (obj.type=N'AF')
251 ORDER BY
252 [Schema] ASC,[Name] ASC
253 go
254 SELECT
255 SCHEMA_NAME(s.schema_id) AS [Schema],
256 s.name AS [Name],
257 s.object_id AS [ID],
258 N'' AS [BaseDatabase],
259 N'' AS [BaseObject],
260 N'' AS [BaseSchema],
261 N'' AS [BaseServer],
262
263 CASE OBJECTPROPERTYEX(s.object_id, 'BaseType')
264 WHEN N'U' THEN 1
265 WHEN N'V' THEN 2
266 WHEN N'P' THEN 3
267 WHEN N'FN' THEN 4
268 WHEN N'TF' THEN 5
269 WHEN N'IF' THEN 6
270 WHEN N'X' THEN 7
271 WHEN N'RF' THEN 8
272 WHEN N'PC' THEN 9
273 WHEN N'FS' THEN 10
274 WHEN N'FT' THEN 11
275 WHEN N'AF' THEN 12 ELSE 0 END
276 AS [BaseType],
277 s.base_object_name AS [BaseObjectName]
278 FROM
279 sys.synonyms AS s
280 ORDER BY
281 [Schema] ASC,[Name] ASC
282 go
283 SELECT
284 SCHEMA_NAME(xproc.schema_id) AS [Schema],
285 xproc.name AS [Name],
286 xproc.object_id AS [ID],
287 CAST(
288 xproc.is_ms_shipped
289 AS bit) AS [IsSystemObject]
290 FROM
291 sys.all_objects AS xproc
292 WHERE
293 (xproc.type='X')
294 ORDER BY

```

```
295 [Schema] ASC,[Name] ASC
296 go
297 SELECT
298 sst.name AS [Schema],
299 st.name AS [Name]
300 FROM
301 sys.types AS st
302 INNER JOIN sys.schemas AS sst ON sst.schema_id = st.schema_id
303 WHERE
304 (st.schema_id!=4 and st.system_type_id!=240 and st.user_type_id != st.
305 system_type_id and st.is_table_type != 1)
306 [Schema] ASC,[Name] ASC
307 go
308 SELECT
309 SCHEMANAME(tt.schema_id) AS [Schema],
310 tt.name AS [Name]
311 FROM
312 sys.table_types AS tt
313 INNER JOIN sys.schemas AS stt ON stt.schema_id = tt.schema_id
314 ORDER BY
315 [Schema] ASC,[Name] ASC
316 go
317 SELECT
318 satypes.name AS [Schema],
319 atypes.name AS [Name]
320 FROM
321 sys.assembly_types AS atypes
322 INNER JOIN sys.assemblies AS asmb1 ON (asmb1.assembly_id = atypes.assembly_id)
323 and (atypes.is_user_defined = 1)
324 INNER JOIN sys.schemas AS satypes ON satypes.schema_id = atypes.schema_id
325 ORDER BY
326 [Schema] ASC,[Name] ASC
327 go
328 SELECT @@SPID;
329 go
330 select * from [dbo].[IMDb$]
331 go
332 use [Netflix Prize Data BETA]
333 go
334 use [Netflix Prize Data BETA]
335 go
336 SELECT @@SPID;
337 go
338 select * from [dbo].[MovieTitles$];
339 go
340 SELECT @@SPID;
341 go
342 select * from [dbo].[TrainingData$];
343 go
344 SELECT @@SPID;
345 go
346
347 SELECT
348 *
349 FROM
350 INFORMATION_SCHEMA.TABLES;
351
```

```

352 go
353 SELECT @@SPID;
354 go
355 SELECT
356     name,
357     crdate
358 FROM
359     SYSOBJECTS
360 WHERE
361     xtype = 'U';
362
363 go
364 SET DEADLOCK_PRIORITY -10
365 go
366 SELECT target_data
367                                     FROM sys.
368 dm_xe_session_targets xet WITH(nolock)
369                                     JOIN sys.
370 dm_xe_sessions xes WITH(nolock)
371                                     ON xes.
372 address = xet.event_session_address
373                                     WHERE xes.
374 name = 'telemetry_xevents'
375                                     AND xet.
376 target_name = 'ring_buffer'
377 go
378 SET DEADLOCK_PRIORITY -10
379 go
380 if not exists (select * from sys.dm_xe_sessions where name = 'telemetry_xevents'
381 ')
382     alter event session telemetry_xevents on server state=start
383 go
384 SELECT
385     dtb.name AS [Name],
386     dtb.database_id AS [ID],
387     CAST(has_dbaccess(dtb.name) AS bit) AS [IsAccessible]
388 FROM
389     master.sys.databases AS dtb
390 ORDER BY
391     [Name] ASC
392 go
393 use [Netflix Prize Data BETA]
394 go
395 use [Netflix Prize Data BETA]
396 go
397 use [Netflix Prize Data BETA]
398 go
399 SELECT @@SPID;
400 go
401 select schema_name(tab.schema_id) as schema_name,
402     tab.name as table_name,
403     col.column_id,
404     col.name as column_name,
405     t.name as data_type,
406     col.max_length,
407     col.precision
408 from sys.tables as tab
409     inner join sys.columns as col
410     on tab.object_id = col.object_id

```

```

405     left join sys.types as t
406     on col.user_type_id = t.user_type_id
407 order by schema_name,
408     table_name,
409     column_id;
410 go
411 SELECT
412 dtb.name AS [Name],
413 dtb.database_id AS [ID],
414 CAST(has_dbaccess(dtb.name) AS bit) AS [IsAccessible]
415 FROM
416 master.sys.databases AS dtb
417 ORDER BY
418 [Name] ASC
419 go
420 SELECT @@SPID;
421 go
422 select * from IMDb$ I, MovieTitles$ M , TrainingData$ T where
423 I.Ratings=T.Grade
424 and M.Title= T.Movies
425 and I.[Date ]= T.[Date of Grade]
426 go
427 SET DEADLOCK_PRIORITY -10
428 go
429 SELECT target_data
430
431 dm_xe_session_targets xet WITH(nolock)
432
433 dm_xe_sessions xes WITH(nolock)
434
435 address = xet.event_session_address
436
437 name = 'telemetry_xevents'
438
439 target_name = 'ring_buffer'
440
441 go
442 SET DEADLOCK_PRIORITY -10
443 go
444 if not exists (select * from sys.dm_xe_sessions where name = 'telemetry_xevents'
445 ')
446     alter event session telemetry_xevents on server state=start
447 go
448 SELECT
449 dtb.name AS [Name],
450 dtb.database_id AS [ID],
451 CAST(has_dbaccess(dtb.name) AS bit) AS [IsAccessible]
452 FROM
453 master.sys.databases AS dtb
454 ORDER BY
455 [Name] ASC
456 go
457 SELECT @@SPID;
458 go
459 select I.[Usernames ], M.Title, T.[User ID] , T.Grade from IMDb$ I,
460 MovieTitles$ M , TrainingData$ T where
461 I.Ratings=T.Grade
462 and M.Title= T.Movies
463 and I.[Date ]= T.[Date of Grade]
464
465

```



```

457 go
458 SELECT @@SPID;
459 go
460 select * from IMDb$ I, MovieTitles$ M , TrainingData$ T where
461 I.Ratings=T.Grade
462 and M.Title= T.Movies
463 and I.[Date ]= T.[Date of Grade];
464 go
465 SELECT
466 dtb.name AS [Name],
467 dtb.database_id AS [ID],
468 CAST(has_dbaccess(dtb.name) AS bit) AS [IsAccessible]
469 FROM
470 master.sys.databases AS dtb
471 ORDER BY
472 [Name] ASC
473 go
474 SELECT @@SPID;
475 go
476 select * from IMDb$ I, MovieTitles$ M , TrainingData$ T where
477 I.Ratings=T.Grade
478 —and M.Title= T.Movies
479 and I.[Date ]= T.[Date of Grade];
480 go
481 SELECT @@SPID;
482 go
483 select distinct I.[Usernames ], M.Title , T.[User ID] , T.Grade from IMDb$ I,
484 MovieTitles$ M , TrainingData$ T where
485 I.Ratings=T.Grade
486 and M.Title= T.Movies
487 and I.[Date ]= T.[Date of Grade]
488 go
489 SET DEADLOCK_PRIORITY -10
490 go
491 SELECT target_data
492 FROM sys.
493 dm_xe_session_targets xet WITH(nolock) JOIN sys.
494 dm_xe_sessions xes WITH(nolock) ON xes.
495 address = xet.event_session_address WHERE xes.
496 name = 'telemetry_xevents' AND xet.
497 target_name = 'ring_buffer'
498 go
499 SET DEADLOCK_PRIORITY -10
500 go
501 if not exists (select * from sys.dm_xe_sessions where name = 'telemetry_xevents
502 ')
503 alter event session telemetry_xevents on server state=start
504 go
505 SELECT
506 dtb.name AS [Name],
507 dtb.database_id AS [ID],
508 CAST(has_dbaccess(dtb.name) AS bit) AS [IsAccessible]
509 FROM
510 master.sys.databases AS dtb

```

```
509 ORDER BY
510 [Name] ASC
511 go
512 SELECT @@SPID;
513 go
514
515 select distinct I.[Usernames ] from IMDB$ I, MovieTitles$ M , TrainingData$ T
    where
516 I.Ratings=T.Grade
517 and M.Title= T.Movies
518 and I.[Date ]= T.[Date of Grade] ;
519 go
520 SELECT @@SPID;
521 go
522 select distinct I.[Usernames ] from IMDB$ I
523 go
524 SELECT @@SPID;
525 go
526 select distinct I.[Usernames ], M.Title , T.[User ID] ,
527             T.Grade from IMDB$ I, MovieTitles$ M , TrainingData$ T where
528 I.Ratings=T.Grade
529 and M.Title= T.Movies
530 and I.[Date ]= T.[Date of Grade] ;
531
532 go
533 SELECT @@SPID;
534 go
535 select distinct T.[User ID] ,
536             from [dbo].[TrainingData$];
537
538 go
539 SELECT @@SPID;
540 go
541 select distinct T.[User ID]
542             from [dbo].[TrainingData$];
543
544 go
545 SELECT @@SPID;
546 go
547 select distinct T.[User ID]
548             from [dbo].[TrainingData$] T;
549
550 go
551 SELECT @@SPID;
552 go
553 select distinct I.[Usernames ], M.Title , T.[User ID] ,
554             T.Grade from IMDB$ I, MovieTitles$ M , TrainingData$ T where
555 I.Ratings=T.Grade
556 and M.Title= T.Movies
557 and I.[Date ]= T.[Date of Grade] ;
558 go
559 SELECT
560 dtb.name AS [Name] ,
561 dtb.database_id AS [ID] ,
562 CAST(has_dbaccess(dtb.name) AS bit) AS [IsAccessible]
563 FROM
564 master.sys.databases AS dtb
565 ORDER BY
566 [Name] ASC
```

```

567 go
568 SET DEADLOCK_PRIORITY -10
569 go
570 SELECT target_data
571                                     FROM sys.
572 dm_xe_session_targets xet WITH(nolock)
573                                     JOIN sys.
574 dm_xe_sessions xes WITH(nolock)
575                                     ON xes.
576 address = xet.event_session_address
577                                     WHERE xes.
578 name = 'telemetry_xevents'
579                                     AND xet.
580 target_name = 'ring_buffer'
581 go
582 SET DEADLOCK_PRIORITY -10
583 go
584 if not exists (select * from sys.dm_xe_sessions where name = 'telemetry_xevents
585 ')
586     alter event session telemetry_xevents on server state=start
587 go
588 SELECT @@SPID;
589 go
590 select distinct I.[Usernames ], M.Title , count(T.[User ID])
591             from IMDB$ I, MovieTitles$ M , TrainingData$ T where
592 I.Ratings=T.Grade
593 and M.Title= T.Movies
594 and I.[Date ]= T.[Date of Grade]
595 group by distinct I.[Usernames ], M.Title;
596 go
597 SELECT @@SPID;
598 go
599 select distinct I.[Usernames ], M.Title , count(T.[User ID])
600             from IMDB$ I, MovieTitles$ M , TrainingData$ T where
601 I.Ratings=T.Grade
602 and M.Title= T.Movies
603 and I.[Date ]= T.[Date of Grade]
604 group by I.[Usernames ], M.Title;
605 go
606 SELECT @@SPID;
607 go
608 select distinct I.[Usernames ], M.Title , count(T.[User ID]) Num_USERID
609             from IMDB$ I, MovieTitles$ M , TrainingData$ T where
610 I.Ratings=T.Grade
611 and M.Title= T.Movies
612 and I.[Date ]= T.[Date of Grade]
613 group by I.[Usernames ], M.Title;
614 go
615 SELECT @@SPID;
616 go
617 select distinct I.[Usernames ], M.Title , count(T.[User ID]) Num_USERID
618             from IMDB$ I, MovieTitles$ M , TrainingData$ T where
619 I.Ratings=T.Grade
620 and M.Title= T.Movies
621 and I.[Date ]= T.[Date of Grade]
622 group by I.[Usernames ], M.Title
623 group by Num_USERID ;
624 go
625 SELECT @@SPID;

```

```
620 go
621 select distinct I.[Usernames ], M.Title , count(T.[User ID]) Num_USERID
622         from  IMDB$ I, MovieTitles$ M , TrainingData$ T where
623 I.Ratings=T.Grade
624 and M.Title= T.Movies
625 and I.[Date ]= T.[Date of Grade]
626 group by  I.[Usernames ], M.Title
627 order by Num_USERID ;
628 go
629 SELECT
630 dtb.name AS [Name],
631 dtb.database_id AS [ID],
632 CAST(has_dbaccess(dtb.name) AS bit) AS [IsAccessible]
633 FROM
634 master.sys.databases AS dtb
635 ORDER BY
636 [Name] ASC
637 go
638 SELECT @@SPID;
639 go
640 select distinct I.[Usernames ], count(T.[User ID]) Num_USERID
641         from  IMDB$ I, MovieTitles$ M , TrainingData$ T where
642 I.Ratings=T.Grade
643 and M.Title= T.Movies
644 and I.[Date ]= T.[Date of Grade]
645 group by  I.[Usernames ], M.Title
646 order by Num_USERID ;
647 go
648 SELECT @@SPID;
649 go
650 select distinct I.[Usernames ], M.Title , T.[User ID] ,
651         T.Grade from  IMDB$ I, MovieTitles$ M , TrainingData$ T where
652 I.Ratings=T.Grade
653 and M.Title= T.Movies
654 and I.[Date ]= T.[Date of Grade] ;
655 go
656 SELECT @@SPID;
657 go
658 select  I.[Usernames ], count(T.[User ID]) Num_USERID
659         from  IMDB$ I, MovieTitles$ M , TrainingData$ T where
660 I.Ratings=T.Grade
661 and M.Title= T.Movies
662 and I.[Date ]= T.[Date of Grade]
663 group by  I.[Usernames ], M.Title
664 order by Num_USERID ;
665 go
666 SELECT @@SPID;
667 go
668
669 select  I.[Usernames ], M.Title , T.[User ID] ,
670         T.Grade from  IMDB$ I, MovieTitles$ M , TrainingData$ T where
671 I.Ratings=T.Grade
672 and M.Title= T.Movies
673 and I.[Date ]= T.[Date of Grade] ;
674 go
675 SELECT
676 dtb.name AS [Name],
677 dtb.database_id AS [ID],
678 CAST(has_dbaccess(dtb.name) AS bit) AS [IsAccessible]
```

```

679 FROM
680 master.sys.databases AS dtb
681 ORDER BY
682 [Name] ASC
683 go
684 SET DEADLOCK_PRIORITY -10
685 go
686 SELECT target_data
687 FROM sys.dm_xe_session_targets xet WITH(nolock)
688 JOIN sys.dm_xe_sessions xes WITH(nolock)
689 ON xes.address = xet.event_session_address
690 WHERE xes.name = 'telemetry_xevents'
691 AND xet.target_name = 'ring_buffer'
692 go
693 SET DEADLOCK_PRIORITY -10
694 go
695 if not exists (select * from sys.dm_xe_sessions where name = 'telemetry_xevents
')
696     alter event session telemetry_xevents on server state=start
697 go
698 SELECT
699 dtb.name AS [Name],
700 dtb.database_id AS [ID],
701 CAST(has_dbaccess(dtb.name) AS bit) AS [IsAccessible]
702 FROM
703 master.sys.databases AS dtb
704 ORDER BY
705 [Name] ASC
706 go
707 SELECT @@SPID;
708 go
709 select * from [dbo].[IMDb$] I where I.[Usernames ] = 'bauer24@hotmail.co.uk';
710
711 go
712 SELECT @@SPID;
713 go
714 select * from [dbo].[TrainingData$] T where T.[User ID] = 'user 18'
715 go
716 SELECT @@SPID;
717 go
718 select * from [dbo].[TrainingData$] T where T.[User ID] = 'user 20'
719 go
720 SELECT @@SPID;
721 go
722 select I.[Usernames ], count(T.[User ID]) Num_USERID
723     from IMDb$ I, MovieTitles$ M , TrainingData$ T where
724 I.Ratings=T.Grade
725 and M.Title= T.Movies
726 and I.[Date ]= T.[Date of Grade]
727 group by I.[Usernames ], M.Title
728 order by Num_USERID ;
729
730 go
731 SELECT @@SPID;
732 go
733 select I.[Usernames ], M.Title , T.[User ID] ,
734     T.Grade from IMDb$ I, MovieTitles$ M , TrainingData$ T where
735 I.Ratings=T.Grade
736 and M.Title= T.Movies

```

```
737 and I.[Date ]= T.[Date of Grade] ;
738 go
739 SELECT
740 dtb.name AS [Name],
741 dtb.database_id AS [ID],
742 CAST(has_dbaccess(dtb.name) AS bit) AS [IsAccessible]
743 FROM
744 master.sys.databases AS dtb
745 ORDER BY
746 [Name] ASC
747 go
748 SET DEADLOCK_PRIORITY -10
749 go
750 SELECT target_data
751 FROM sys.dm_xe_session_targets xet WITH(nolock)
752 JOIN sys.dm_xe_sessions xes WITH(nolock)
753 ON xes.address = xet.event_session_address
754 WHERE xes.name = 'telemetry_xevents'
755 AND xet.target_name = 'ring_buffer'
756 go
757 SET DEADLOCK_PRIORITY -10
758 go
759 if not exists (select * from sys.dm_xe_sessions where name = 'telemetry_xevents
760 ')
761 alter event session telemetry_xevents on server state=start
762 go
```

Appendices B.1 and B.2 are one sample each from the Normal query dataset and Re-identification query dataset respectively. The SQL query dataset for Appendix B in its entirety is available at [73]

Appendix C

Python Implementation

C.1 Data Preparation and Labelling

```
# regex pattern to add space around special characters
space_adding_symbols_pat = re.compile(r"([@[\],=#\+\-\*/|;\\".()!%<>])")

# reading through each row of the csv file and splitting contents into words
def read_data_and_label(filePath):
    # reading file
    data = pd.read_csv(filePath)
    # setting class label to 0 if normal query and 1 for attack attempt queries
    label = 0 if "normal" in filePath else 1
    list_of_words=[]
    # looping through each row of the csv file
    for index, row in data.iterrows():
        # reading through the columns
        for col in data.columns:
            # getting the cell data
            s = row[col]
            # if data is valid (a word and not an empty space)
            if not pd.isnull(s):
                # converting to string
                s=str(s)
                # adding space around special characters
                spaced = space_adding_symbols_pat.sub(" \\1 ", s)
                # splitting row into words and adding the list of words to dataset
                list_of_words+=spaced.split()

    clean_words=[]
    for word in list_of_words:
        if bool(word and not word.isspace()):
            word=word.strip("")
            # replacing numbers with single token "<NUM>"
            word = "<NUM>" if word.isdigit() else word
```

```
        clean_words.append(word)
    return (clean_words,label)

# looping through each csv file and creating the dataset
rootdir = "/content/sqldataset"
X,Y = [],[]
c=0
for subdir, dirs, files in os.walk(rootdir):
    for file in files:
        #print os.path.join(subdir, file)
        filepath = subdir + os.sep + file
        if filepath.endswith(".csv"):
            print(filepath)
            words,label = read_data_and_label(filepath)
            X.append(words)
            Y.append(label)
            c+=1

data = (X,Y)
# saving the dataset for future usage in pickled format
pickle.dump( data, open( "preprocessed_dataset.p", "wb" ) )
```

C.1.1 Random Word Masking Data Augmentation

```
#Loading previously preprocessed data
data = pickle.load( open( "/content/preprocessed_dataset.p", "rb" ) )

#setting the sequence length for slicing
fixed_length = 1000
#list new variables to store augmented data and their labels
aug_data, new_label = [],[]

#looping through each document
for index, doc in enumerate(data[0]):
    current_range = 0
    #getting total words in the document
    total_doc_words = len(doc)
    #getting the label for the document
    label = data[1][index]

    #slicing the full document into a slice of 1000 words
    while current_range < total_doc_words:
        # if range is less than 1000 words, then break
        if total_doc_words - current_range < fixed_length:
            break
        #getting the 1000 sliced words
        sliced_word_list = doc[current_range:(current_range + fixed_length)]
```



```

#adding the sliced words to the dataset (using newly defined variables)
aug_data.append(sliced_word_list)
new_label.append(label)

#creating 100 new data points with length 1000
for i in range(100):
    #randomly generating 5% indices to mask some words
    random_list = random.sample(range(0, fixed_length), int(fixed_length*0.05))
    #first, copying all words as they are
    new_data = sliced_word_list.copy()

    #looping through the indices and replacing some random words with the <MASK> token
    for idx in random_list:
        new_data[idx] = "<MASK>"

    #adding the newly masked data and its label into the augmented dataset list
    aug_data.append(new_data)
    new_label.append(label)
    #current_range is updated to point to the next 1000 words range
    current_range += fixed_length

print(total_doc_words)

```

C.1.2 Word Cloud for Random Word Masking Data

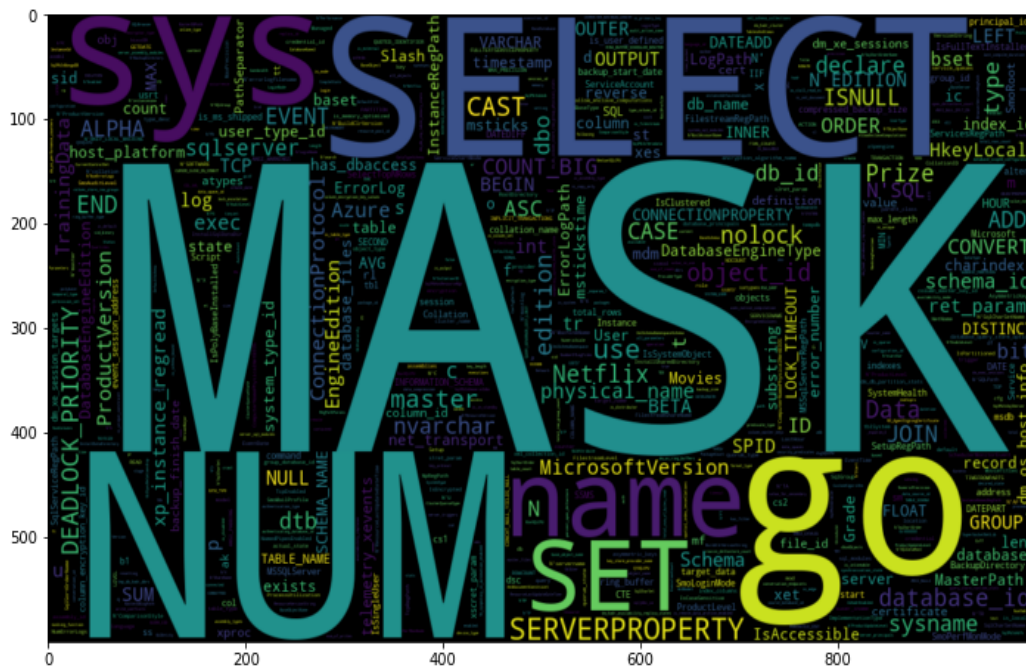


Figure C.1: Word cloud for the normal class


```
#create dataframe
tf_idf_dataframe=pd.DataFrame(tf_idf_data.toarray(),columns=tfidf.get_feature_names())
tf_idf_dataframe

# save tfIdf vectorizer
pickle.dump(tfidf, open("Tfidf.pickle", "wb"))
```


Appendix D

Classification Experiment

D.1 Random Word Masking with *Tokenizer*

D.1.1 MLP Model Parameters

```
Model: "sequential"
-----
Layer (type)                Output Shape         Param #
-----
dense (Dense)                (None, 500)         871500
dense_1 (Dense)              (None, 1)           501
-----
Total params: 872,001
Trainable params: 872,001
Non-trainable params: 0
-----
```

Figure D.1: MLP model parameters

D.1.2 MLP Model Training

```

Epoch 1/10
187/187 - 4s - loss: 0.5204 - accuracy: 0.7420 - val_loss: 0.3544 - val_accuracy: 0.8496 - 4s/epoch - 19ms/step
Epoch 2/10
187/187 - 2s - loss: 0.2625 - accuracy: 0.9174 - val_loss: 0.2052 - val_accuracy: 0.9416 - 2s/epoch - 10ms/step
Epoch 3/10
187/187 - 2s - loss: 0.1702 - accuracy: 0.9432 - val_loss: 0.1505 - val_accuracy: 0.9496 - 2s/epoch - 9ms/step
Epoch 4/10
187/187 - 2s - loss: 0.1325 - accuracy: 0.9541 - val_loss: 0.1295 - val_accuracy: 0.9657 - 2s/epoch - 9ms/step
Epoch 5/10
187/187 - 2s - loss: 0.1140 - accuracy: 0.9604 - val_loss: 0.1109 - val_accuracy: 0.9530 - 2s/epoch - 9ms/step
Epoch 6/10
187/187 - 2s - loss: 0.1038 - accuracy: 0.9627 - val_loss: 0.0985 - val_accuracy: 0.9563 - 2s/epoch - 9ms/step
Epoch 7/10
187/187 - 2s - loss: 0.0883 - accuracy: 0.9735 - val_loss: 0.0873 - val_accuracy: 0.9644 - 2s/epoch - 9ms/step
Epoch 8/10
187/187 - 2s - loss: 0.0793 - accuracy: 0.9788 - val_loss: 0.0774 - val_accuracy: 0.9899 - 2s/epoch - 10ms/step
Epoch 9/10
187/187 - 2s - loss: 0.0714 - accuracy: 0.9805 - val_loss: 0.0711 - val_accuracy: 0.9698 - 2s/epoch - 10ms/step
Epoch 10/10
187/187 - 2s - loss: 0.0628 - accuracy: 0.9827 - val_loss: 0.0617 - val_accuracy: 0.9852 - 2s/epoch - 9ms/step

```

Figure D.2: MLP model training

D.1.3 MLP classification report

```

=== Classification Report ===

```

	precision	recall	f1-score	support
0	0.97	1.00	0.99	1854
1	1.00	0.97	0.99	1813
accuracy			0.99	3667
macro avg	0.99	0.99	0.99	3667
weighted avg	0.99	0.99	0.99	3667

```

=== Evaluation Metrics ===
Accuracy for Test : 0.986910280883556
Precision for Test : 0.9872406207981667
Recall for Test : 0.986910280883556
F1-Score for Test : 0.9869061199033331
AUC score for Test : 0.9867622724765581

```

Figure D.3: MLP classification report

D.1.4 Gaussian NB classification report

```

=== Classification Report ===
              precision    recall  f1-score   support

     0         1.00      0.56      0.72     1123
     1         0.69      1.00      0.82     1099

   accuracy          0.85          0.78          0.78     2222
  macro avg          0.85          0.78          0.77     2222
 weighted avg          0.85          0.78          0.77     2222

=== Evaluation Metrics ===

Accuracy for Test : 0.7781278127812782
Precision for Test : 0.8468357200041612
Recall for Test : 0.7781278127812782
F1-Score for Test : 0.7672528691931136
AUC score for Test : 0.7804986642920748

```

Figure D.4: Gaussian NB classification report

D.1.5 Bernoulli NB classification report

```

=== Classification Report ===
              precision    recall  f1-score   support

     0         0.88      0.68      0.77     1123
     1         0.74      0.90      0.81     1099

   accuracy          0.81          0.79          0.79     2222
  macro avg          0.81          0.79          0.79     2222
 weighted avg          0.81          0.79          0.79     2222

=== Evaluation Metrics ===

Accuracy for Test : 0.7916291629162916
Precision for Test : 0.806982687875849
Recall for Test : 0.7916291629162916
F1-Score for Test : 0.7893212217223579
AUC score for Test : 0.7927959279746746

```

Figure D.5: Bernoulli NB classification report

D.1.6 Categorical NB classification report

```
=== Classification Report ===
              precision    recall  f1-score   support

     0           0.00      0.00      0.00     1123
     1           0.49      1.00      0.66     1099

 accuracy              0.49     2222
 macro avg           0.25      0.50      0.33     2222
 weighted avg       0.24      0.49      0.33     2222

=== Evaluation Metrics ===

Accuracy for Test : 0.4945994599459946
Precision for Test : 0.2446286257788695
Recall for Test : 0.4945994599459946
F1-Score for Test : 0.3273500791813599
AUC score for Test : 0.5
```

Figure D.6: Categorical NB classification report

D.1.7 Baseline Multinomial NB classification report

```
=== Classification Report ===
              precision    recall  f1-score   support

     0           0.93      0.74      0.82     1123
     1           0.78      0.94      0.85     1099

 accuracy              0.84     2222
 macro avg           0.85      0.84      0.84     2222
 weighted avg       0.85      0.84      0.84     2222

=== Evaluation Metrics ===

Accuracy for Test : 0.8370837083708371
Precision for Test : 0.8522131417294416
Recall for Test : 0.8370837083708371
F1-Score for Test : 0.8355358986426742
AUC score for Test : 0.8381731307583921
```

Figure D.7: Baseline Multinomial NB classification report

D.1.8 Tuned Multinomial NB classification report

```

=== Classification Report ===
              precision    recall  f1-score   support

     0         0.94         0.74         0.83        1123
     1         0.78         0.95         0.86        1099

 accuracy          0.86
 macro avg          0.86
 weighted avg       0.86

=== Evaluation Metrics ===

Accuracy for Test : 0.8442844284428442
Precision for Test : 0.860561462399378
Recall for Test : 0.8442844284428442
F1-Score for Test : 0.842720648461074
AUC score for Test : 0.8454038602242628

```

Figure D.8: Tuned Multinomial NB classification report

D.1.9 KNN with Euclidean Distance

```

=== Classification Report ===
              precision    recall  f1-score   support

     0         1.00         1.00         1.00        1123
     1         1.00         1.00         1.00        1099

 accuracy          1.00
 macro avg          1.00
 weighted avg       1.00

=== Evaluation Metrics ===

Accuracy for Test : 0.9995499549954996
Precision for Test : 0.999550355391411
Recall for Test : 0.9995499549954996
F1-Score for Test : 0.9995499527164025
AUC score for Test : 0.9995450409463149

```

Figure D.9: KNN with Euclidean Distance classification report

D.1.10 KNN with Manhattan Distance

```
=== Classification Report ===
              precision    recall  f1-score   support

     0           1.00       1.00       1.00        1123
     1           1.00       1.00       1.00        1099

 accuracy          1.00
macro avg          1.00       1.00       1.00        2222
weighted avg          1.00       1.00       1.00        2222

=== Evaluation Metrics ===

Accuracy for Test : 1.0
Precision for Test : 1.0
Recall for Test : 1.0
F1-Score for Test : 1.0
AUC score for Test : 1.0
```

Figure D.10: KNN with Manhattan Distance classification report

D.1.11 Baseline Logistic Regression

```
=== Classification Report ===
              precision    recall  f1-score   support

     0           0.89       0.95       0.92        1123
     1           0.95       0.88       0.91        1099

 accuracy          0.92
macro avg          0.92       0.92       0.92        2222
weighted avg          0.92       0.92       0.92        2222

=== Evaluation Metrics ===

Accuracy for Test : 0.9158415841584159
Precision for Test : 0.9180008361062986
Recall for Test : 0.9158415841584159
F1-Score for Test : 0.915696287833815
AUC score for Test : 0.9154379801276479
```

Figure D.11: Baseline Logistic Regression classification report

D.1.12 Tuned Logistic Regression

```

=== Classification Report ===

              precision    recall  f1-score   support

     0           1.00       1.00       1.00        1123
     1           1.00       1.00       1.00        1099

   accuracy                1.00
  macro avg                1.00
 weighted avg                1.00

=== Evaluation Metrics ===

Accuracy for Test : 1.0
Precision for Test : 1.0
Recall for Test : 1.0
F1-Score for Test : 1.0
AUC score for Test : 1.0

```

Figure D.12: Tuned Logistic Regression classification report

D.2 Random Token Insertion with *Tokenizer*

D.2.1 MLP Model Parameters

```

Epoch 1/10
124/124 - 4s - loss: 0.5665 - accuracy: 0.7067 - val_loss: 0.4548 - val_accuracy: 0.7419 - 4s/epoch - 36ms/step
Epoch 2/10
124/124 - 4s - loss: 0.4194 - accuracy: 0.7770 - val_loss: 0.3872 - val_accuracy: 0.8249 - 4s/epoch - 31ms/step
Epoch 3/10
124/124 - 5s - loss: 0.3619 - accuracy: 0.8211 - val_loss: 0.3369 - val_accuracy: 0.8492 - 5s/epoch - 40ms/step
Epoch 4/10
124/124 - 2s - loss: 0.3179 - accuracy: 0.8562 - val_loss: 0.2923 - val_accuracy: 0.8836 - 2s/epoch - 17ms/step
Epoch 5/10
124/124 - 2s - loss: 0.2784 - accuracy: 0.8889 - val_loss: 0.2559 - val_accuracy: 0.9059 - 2s/epoch - 16ms/step
Epoch 6/10
124/124 - 2s - loss: 0.2442 - accuracy: 0.9079 - val_loss: 0.2254 - val_accuracy: 0.9413 - 2s/epoch - 16ms/step
Epoch 7/10
124/124 - 2s - loss: 0.2162 - accuracy: 0.9347 - val_loss: 0.1992 - val_accuracy: 0.9393 - 2s/epoch - 17ms/step
Epoch 8/10
124/124 - 3s - loss: 0.1931 - accuracy: 0.9496 - val_loss: 0.1802 - val_accuracy: 0.9494 - 3s/epoch - 20ms/step
Epoch 9/10
124/124 - 3s - loss: 0.1740 - accuracy: 0.9567 - val_loss: 0.1569 - val_accuracy: 0.9706 - 3s/epoch - 24ms/step
Epoch 10/10
124/124 - 2s - loss: 0.1568 - accuracy: 0.9689 - val_loss: 0.1432 - val_accuracy: 0.9706 - 2s/epoch - 16ms/step

```

Figure D.13: MLP model parameters

D.2.2 MLP Model Training

Model: "sequential"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 500)	943000
dense_1 (Dense)	(None, 1)	501

=====
 Total params: 943,501
 Trainable params: 943,501
 Non-trainable params: 0
 =====

Figure D.14: MLP model training

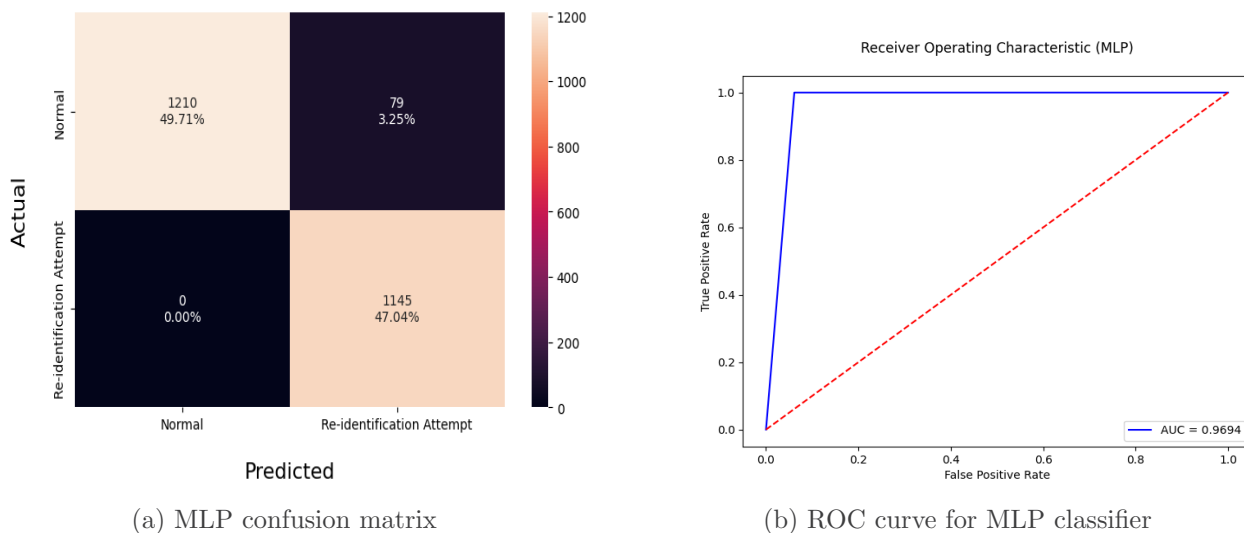


Figure D.15: MLP classification performance evaluation confusion matrix and ROC curve

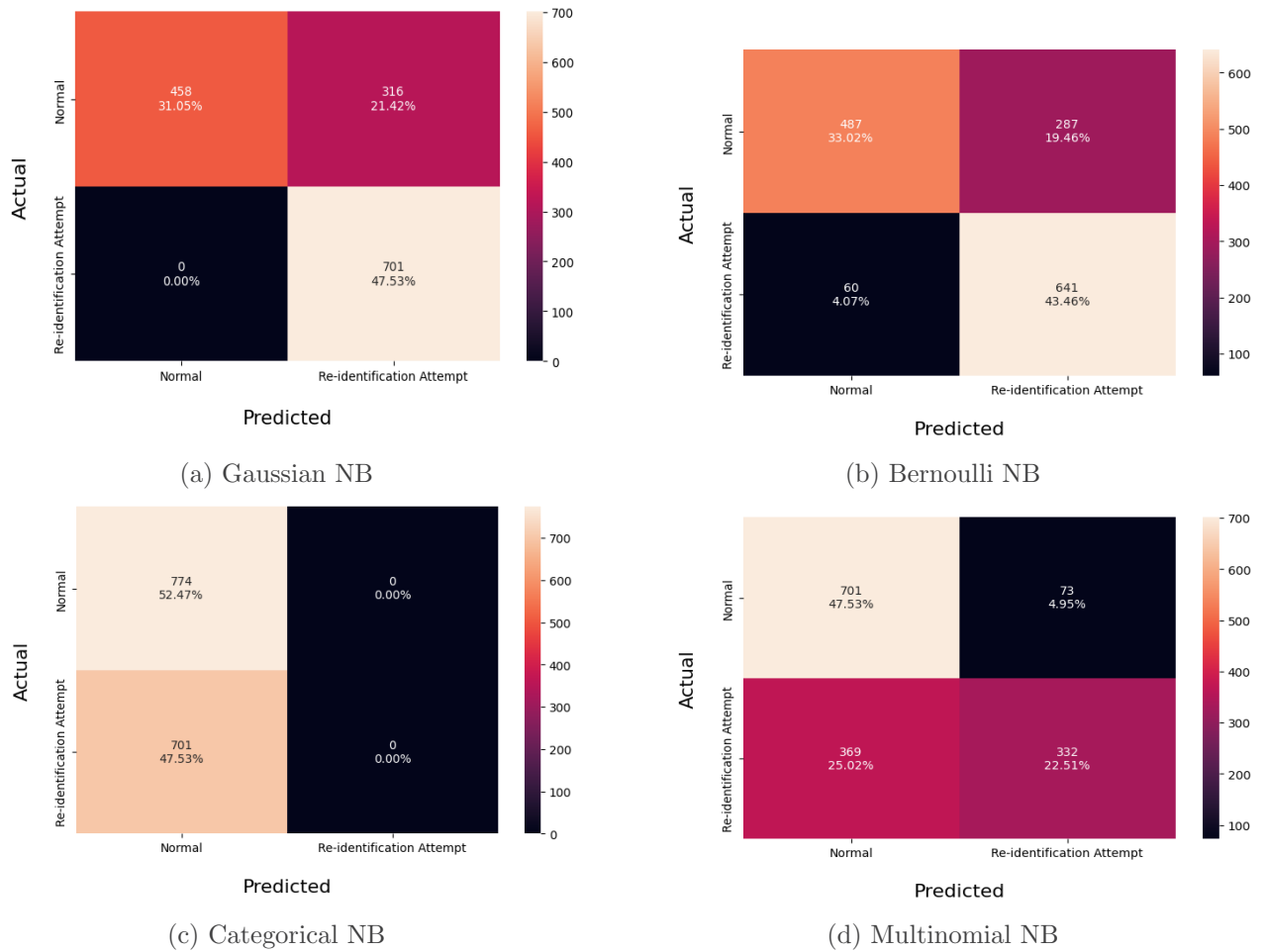


Figure D.16: Classification performance evaluation confusion matrices for different NB classifiers

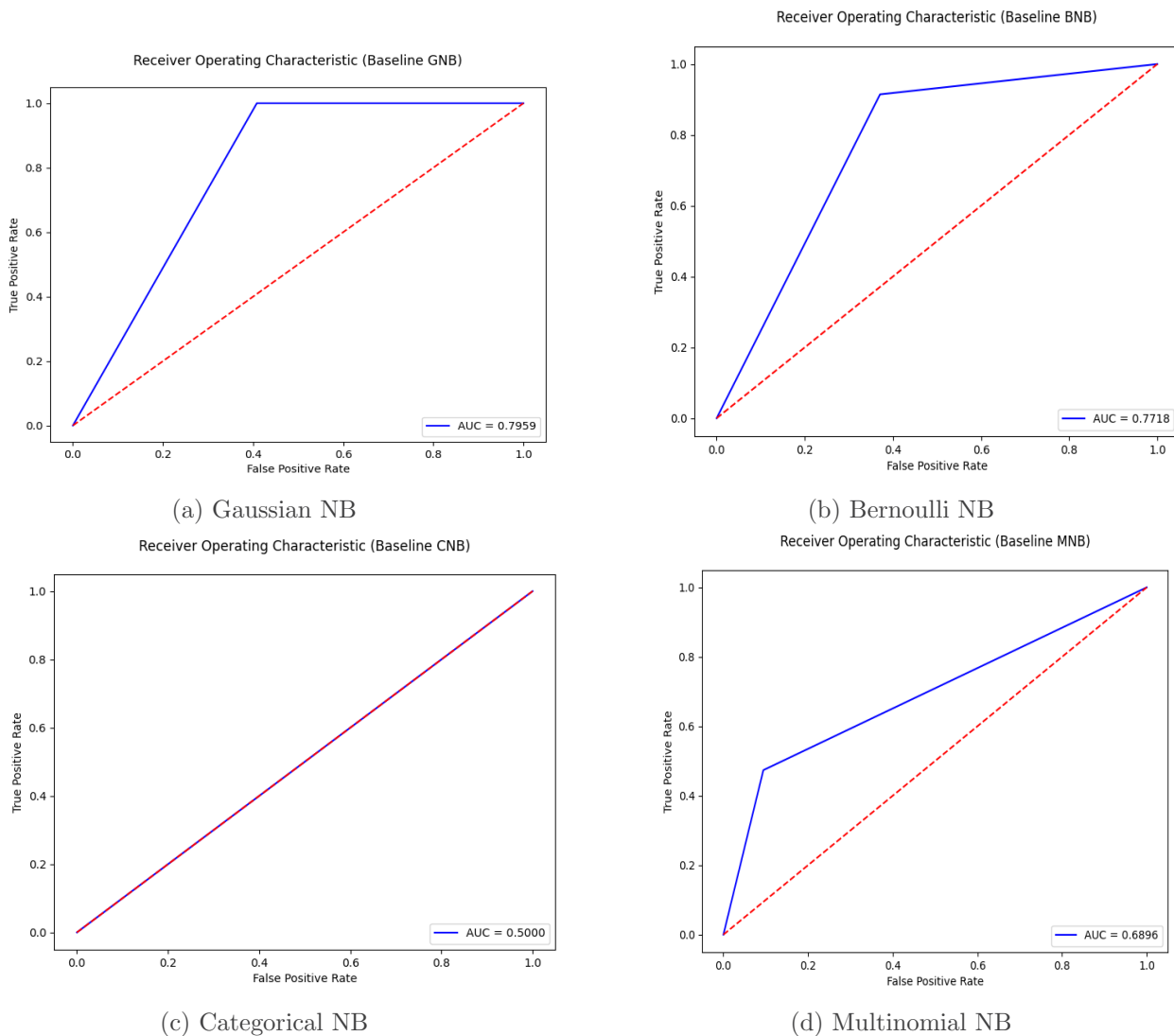
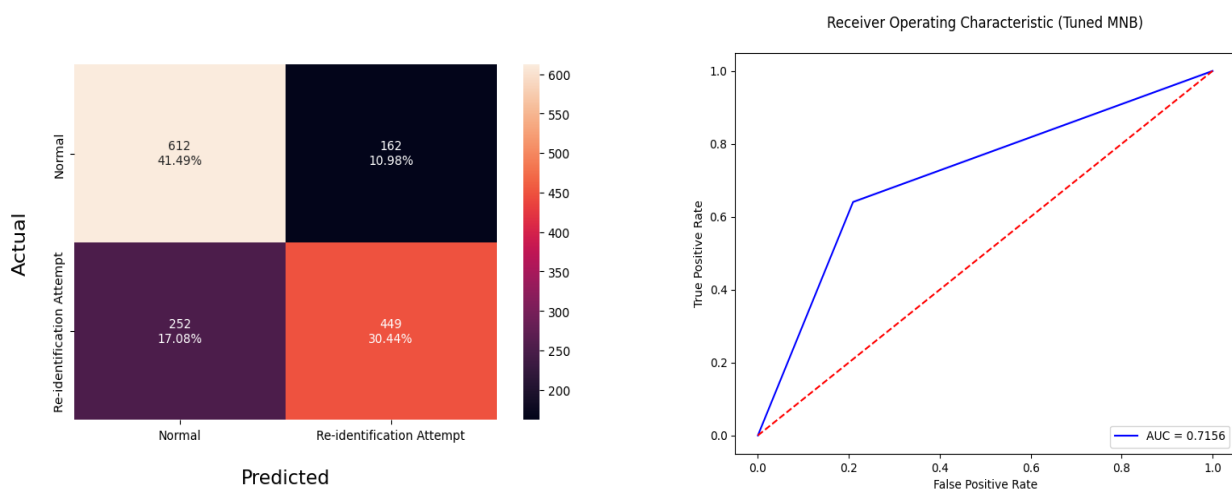


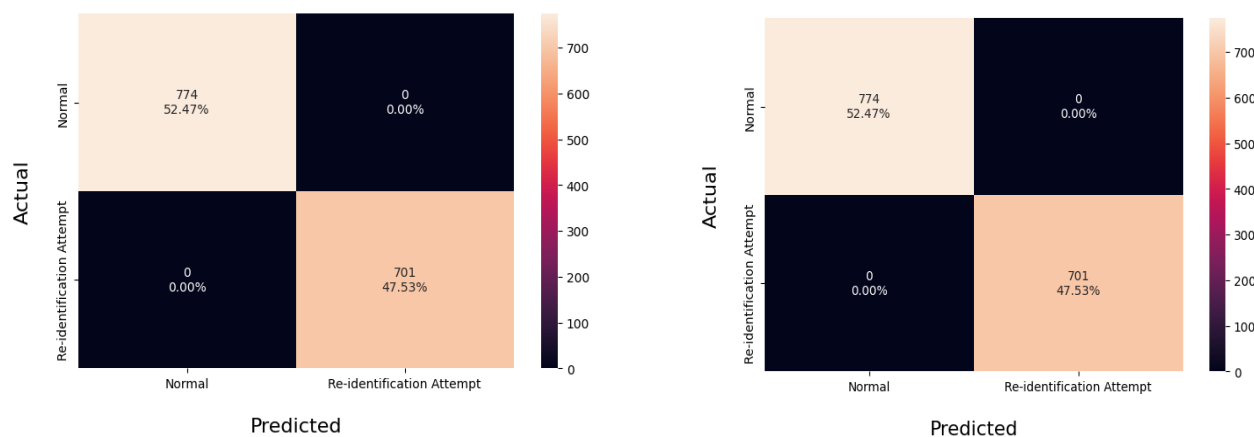
Figure D.17: ROC curves for different NB classifiers



(a) Tuned MNB Confusion Matrix

(b) ROC curve for tuned MNB classifier

Figure D.18: Tuned MNB classification performance evaluation confusion matrix and ROC curve



(a) KNN with Euclidean distance

(b) KNN with Manhattan distance

Figure D.19: Classification performance evaluation confusion matrices for KNN with different distance metric

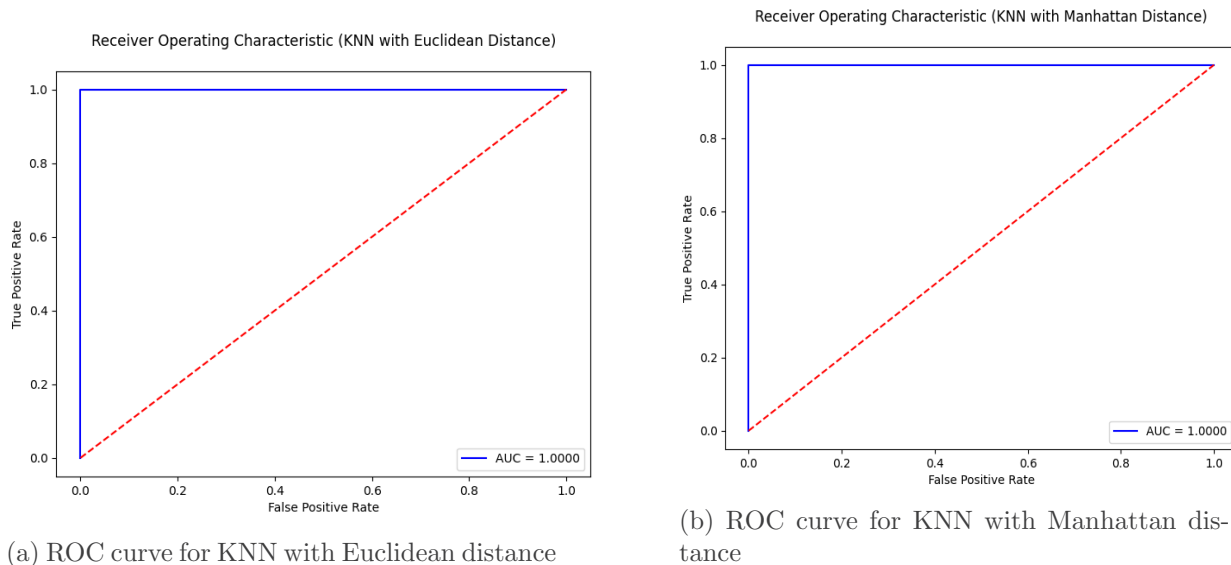


Figure D.20: ROC curves for KNN with different distance metric

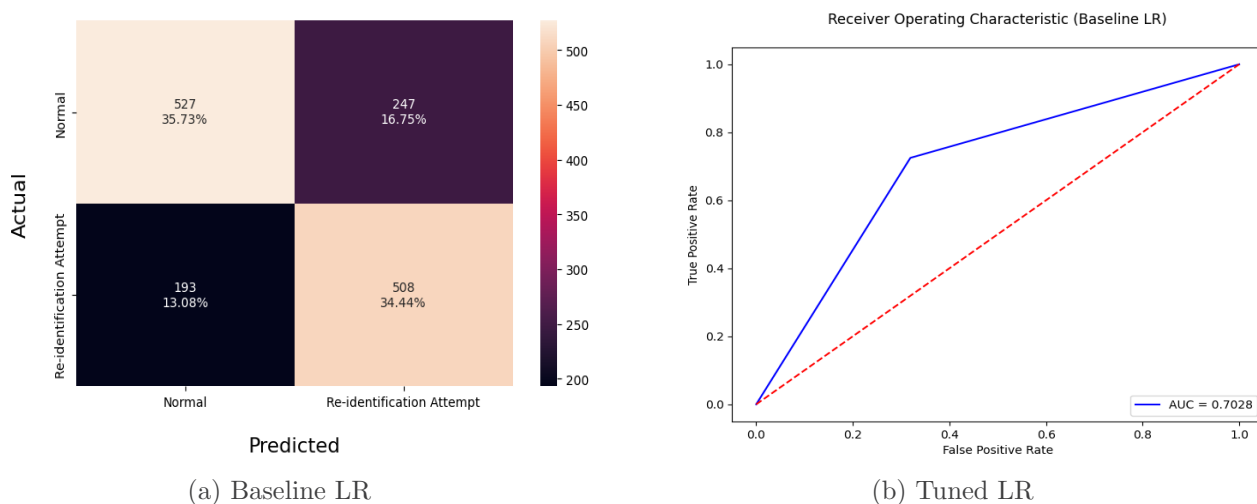
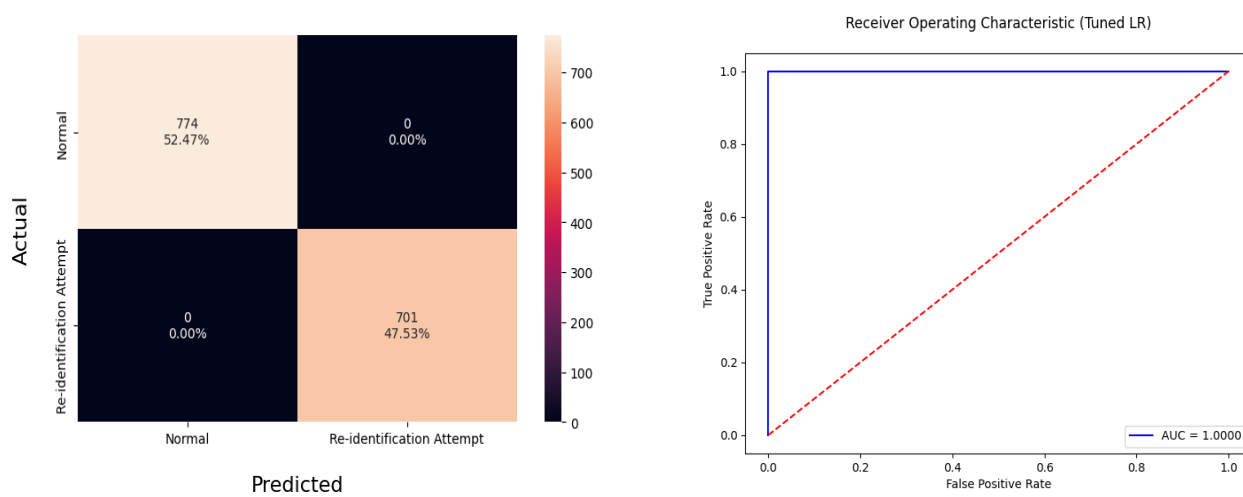


Figure D.21: Classification performance evaluation confusion matrices for baseline and tuned Logistic Regression



(a) ROC curve for baseline LR

(b) ROC curve for tuned LR

Figure D.22: ROC curves for baseline and tuned LR classifiers

D.3 Random Word Masking with *TF-IDF*

D.3.1 MLP Model Parameters

```
Epoch 1/10
187/187 - 4s - loss: 0.3749 - accuracy: 0.8270 - val_loss: 0.2640 - val_accuracy: 0.8496 - 4s/epoch - 19ms/step
Epoch 2/10
187/187 - 2s - loss: 0.1884 - accuracy: 0.9444 - val_loss: 0.1405 - val_accuracy: 0.9691 - 2s/epoch - 11ms/step
Epoch 3/10
187/187 - 2s - loss: 0.1010 - accuracy: 0.9859 - val_loss: 0.0780 - val_accuracy: 0.9987 - 2s/epoch - 11ms/step
Epoch 4/10
187/187 - 2s - loss: 0.0565 - accuracy: 0.9965 - val_loss: 0.0513 - val_accuracy: 0.9893 - 2s/epoch - 12ms/step
Epoch 5/10
187/187 - 3s - loss: 0.0345 - accuracy: 0.9993 - val_loss: 0.0279 - val_accuracy: 1.0000 - 3s/epoch - 15ms/step
Epoch 6/10
187/187 - 3s - loss: 0.0219 - accuracy: 1.0000 - val_loss: 0.0198 - val_accuracy: 1.0000 - 3s/epoch - 17ms/step
Epoch 7/10
187/187 - 2s - loss: 0.0145 - accuracy: 1.0000 - val_loss: 0.0147 - val_accuracy: 1.0000 - 2s/epoch - 12ms/step
Epoch 8/10
187/187 - 2s - loss: 0.0104 - accuracy: 1.0000 - val_loss: 0.0097 - val_accuracy: 1.0000 - 2s/epoch - 11ms/step
Epoch 9/10
187/187 - 2s - loss: 0.0075 - accuracy: 1.0000 - val_loss: 0.0071 - val_accuracy: 1.0000 - 2s/epoch - 11ms/step
Epoch 10/10
187/187 - 2s - loss: 0.0057 - accuracy: 1.0000 - val_loss: 0.0067 - val_accuracy: 1.0000 - 2s/epoch - 12ms/step
```

Figure D.23: MLP model parameters

D.3.2 MLP Model Training

```

Model: "sequential"
-----
Layer (type)                Output Shape                Param #
-----
dense (Dense)                (None, 500)                 820000
dense_1 (Dense)              (None, 1)                   501
-----
Total params: 820,501
Trainable params: 820,501
Non-trainable params: 0
-----

```

Figure D.24: MLP model training

D.3.3 MLP classification report

```

=== Classification Report ===
              precision    recall  f1-score   support

     0         1.00      1.00      1.00     1854
     1         1.00      1.00      1.00     1813

   accuracy          1.00
  macro avg          1.00
 weighted avg          1.00

=== Evaluation Metrics ===
Accuracy for Test : 1.0
Precision for Test : 1.0
Recall for Test : 1.0
F1-Score for Test : 1.0
AUC score for Test : 1.0

```

Figure D.25: MLP classification report

D.3.4 Gaussian NB classification report

```
=== Classification Report ===
              precision    recall  f1-score   support

     0           1.00       0.56      0.72       1123
     1           0.69       1.00      0.82       1099

 accuracy          0.78       2222
 macro avg         0.85       0.78       0.77       2222
 weighted avg     0.85       0.78       0.77       2222

=== Evaluation Metrics ===
Accuracy for Test : 0.7785778577857786
Precision for Test : 0.8470503241398936
Recall for Test : 0.7785778577857786
F1-Score for Test : 0.7677722267316023
AUC score for Test : 0.7809439002671416
```

Figure D.26: Gaussian NB classification report

D.3.5 Bernoulli NB classification report

```
=== Classification Report ===
              precision    recall  f1-score   support

     0           0.86       0.69       0.76       1123
     1           0.73       0.89       0.80       1099

 accuracy          0.79       2222
 macro avg         0.80       0.79       0.78       2222
 weighted avg     0.80       0.79       0.78       2222

=== Evaluation Metrics ===
Accuracy for Test : 0.7853285328532853
Precision for Test : 0.7984286099932869
Recall for Test : 0.7853285328532853
F1-Score for Test : 0.7832698568924827
AUC score for Test : 0.786416778144464
```

Figure D.27: Bernoulli NB classification report

D.3.6 Categorical NB classification report

```

=== Classification Report ===
              precision    recall  f1-score   support

     0       0.00         0.00         0.00         1123
     1       0.49         1.00         0.66         1099

 accuracy          0.49         0.49         0.49         2222
 macro avg         0.25         0.50         0.33         2222
 weighted avg      0.24         0.49         0.33         2222

=== Evaluation Metrics ===

Accuracy for Test : 0.4945994599459946
Precision for Test : 0.2446286257788695
Recall for Test : 0.4945994599459946
F1-Score for Test : 0.3273500791813599
AUC score for Test : 0.5

```

Figure D.28: Categorical NB classification report

D.3.7 Baseline Multinomial NB classification report

```

=== Classification Report ===
              precision    recall  f1-score   support

     0       0.87         0.67         0.76         1123
     1       0.73         0.89         0.80         1099

 accuracy          0.78         0.78         0.78         2222
 macro avg         0.80         0.78         0.78         2222
 weighted avg      0.80         0.78         0.78         2222

=== Evaluation Metrics ===

Accuracy for Test : 0.7826282628262826
Precision for Test : 0.7983100389549658
Recall for Test : 0.7826282628262826
F1-Score for Test : 0.7800933036502807
AUC score for Test : 0.7838231469230104

```

Figure D.29: Baseline Multinomial NB classification report

D.3.8 Tuned Multinomial NB classification report

```

=== Classification Report ===
              precision    recall  f1-score   support

     0         0.89         0.67         0.77         1123
     1         0.73         0.92         0.81         1099

 accuracy          0.79         0.79         0.79         2222
 macro avg         0.81         0.79         0.79         2222
 weighted avg         0.81         0.79         0.79         2222

=== Evaluation Metrics ===

Accuracy for Test : 0.7934293429342935
Precision for Test : 0.813059637717912
Recall for Test : 0.7934293429342935
F1-Score for Test : 0.7905040840595091
AUC score for Test : 0.7947421642114543

```

Figure D.30: Tuned Multinomial NB classification report

D.3.9 KNN with Euclidean Distance

```

=== Classification Report ===
              precision    recall  f1-score   support

     0         1.00         1.00         1.00         1123
     1         1.00         1.00         1.00         1099

 accuracy          1.00         1.00         1.00         2222
 macro avg         1.00         1.00         1.00         2222
 weighted avg         1.00         1.00         1.00         2222

=== Evaluation Metrics ===

Accuracy for Test : 0.9990999099909991
Precision for Test : 0.9990999099909991
Recall for Test : 0.9990999099909991
F1-Score for Test : 0.9990999099909991
AUC score for Test : 0.9990998049712482

```

Figure D.31: KNN with Euclidean Distance classification report

D.3.10 KNN with Manhattan Distance

```

=== Classification Report ===
              precision    recall  f1-score   support

     0           1.00         1.00         1.00         1123
     1           1.00         1.00         1.00         1099

   accuracy                1.00         2222
  macro avg           1.00         1.00         1.00         2222
 weighted avg           1.00         1.00         1.00         2222

=== Evaluation Metrics ===

Accuracy for Test : 1.0
Precision for Test : 1.0
Recall for Test : 1.0
F1-Score for Test : 1.0
AUC score for Test : 1.0

```

Figure D.32: KNN with Manhattan Distance classification report

D.3.11 Baseline Logistic Regression

```

=== Classification Report ===
              precision    recall  f1-score   support

     0           0.94         0.87         0.91         1123
     1           0.88         0.94         0.91         1099

   accuracy                0.91         2222
  macro avg           0.91         0.91         0.91         2222
 weighted avg           0.91         0.91         0.91         2222

=== Evaluation Metrics ===

Accuracy for Test : 0.9077407740774077
Precision for Test : 0.9099894347253783
Recall for Test : 0.9077407740774077
F1-Score for Test : 0.907654442941515
AUC score for Test : 0.9081237942369693

```

Figure D.33: Baseline Logistic Regression classification report

D.3.12 Tuned Logistic Regression

```
=== Classification Report ===
              precision    recall  f1-score   support

     0           1.00       1.00       1.00       1123
     1           1.00       1.00       1.00       1099

 accuracy          1.00
 macro avg         1.00       1.00       1.00
 weighted avg     1.00       1.00       1.00

=== Evaluation Metrics ===

Accuracy for Test : 1.0
Precision for Test : 1.0
Recall for Test : 1.0
F1-Score for Test : 1.0
AUC score for Test : 1.0
```

Figure D.34: Tuned Logistic Regression classification report