12-17-2019 2:30 PM

Design and Implementation of Anomaly Detections for User Authentication Framework

Iman Abu Sulayman, *The University of Western Ontario*

Supervisor: Ouda, Abdelkader, *The University of Western Ontario*

A thesis submitted in partial fulfillment of the requirements for the Master of Engineering Science degree in Electrical and Computer Engineering

Abstract

Anomaly detection is quickly becoming a very significant tool for a variety of applications such as intrusion detection, fraud detection, fault detection, system health monitoring, and event detection in IoT devices. An application that lacks a strong implementation for anomaly detection is user trait modeling for user authentication purposes. User trait models expose up-to-date representation of the user so that changes in their interests, their learning progress or interactions with the system are noticed and interpreted. The reason behind the lack of adoption in user trait modeling arises from the need of a continuous flow of high-volume data, that is not available in most cases, to achieve high-accuracy detection. This research provides new insight into anomaly detection techniques through Big Data utilization. Three classification approaches are presented for anomaly detection techniques that are aligned with Big Data characteristics: volume, variety and velocity. The classification is supported by applications of machine learning techniques, such as K-means, Hidden Markov Model, Gaussian Distribution and Auto-encoder neural network, with an aim to recommend best techniques to model user behaviour in an adaptive environment. An ingenious implementation of machine learning techniques has been presented that automatically and accurately builds a unique pattern of the users' behaviour. With Big Data characteristics, anomaly detection techniques have become more suitable tools for user trait modeling. A solution model is designed and implemented based on anomaly detection outcomes utilizing user traits for an existing user authentication framework. User traits will be modeled by creating a security user profile for each individual user. This profile is structured and developed to be a seed for a strong real-time user authentication method. The implementation comprises four main steps: prediction of rare user actions, filter security potential actions, build/update user profile, and generate a real-time (i.e., just in time) set of challenging questions. Real-world scenarios have been given showing the benefits of these challenging questions in building secure knowledge-based user authentication systems.

Keywords

User trait Modeling, Big Data, Anomaly Detection, K-means, Gaussian Distribution, Neural Network, User Authentication

Dedication

This thesis is dedicated to my Parents who have given me support throughout every step of my educational career. Also, I dedicate this thesis to my brother and sisters who encouraged me to finish my studies and helped me become the man I am today. Finally, I dedicate this thesis to my wife and sons, who have lived with me during my schooling and support me through my life and encourage me to achieve more.

Acknowledgments

All thanks go to God (Allah) Who has given me the ability to achieve my goals and Guides me to seek knowledge. Allah is the only one who protects me during my study journey, and He is the only source of all of my achievements.

Secondly, I would like to thank my lovely parents for their daily supplications for me, wishing me all the best in my entire life. Thirdly, a sincere thanks to my lovely family (my wife and my two sons) for their patience, support and encouragement through my studies.

I would like to express my deep appreciation to my supervisor Dr. Abdelkader Ouda for his insightful guidance, invaluable advice, and constructive criticism during my MESc. program. His academic expertise helped me to improve my research skills. His support and motivation gave me the confidence and the strength to accomplish my goals.

This work was partially supported by Tiaf University in Saudi Arabian through the Cultural Bureau of Saudi Arabia in Canada. This support is greatly appreciated

Table of Contents

Abstract i

Dedication ii

Acknowledgments iii

Table of Contents iv

List of Tables vii

List of Figures xi

List of Appendices xiv

List of Abbreviations xvii

Chapter 1 1

1 Introduction 1

1.1 Research Motivation 2

1.2 Research Objectives 3

1.3 Research Methodology 4

1.4 Research Contribution 6

1.5 Research Outline 7

Chapter 2 8

2 Literature Review and Background 8

2.1 Literature Review 8

2.2 Anomaly Detection Techniques 11

2.2.1 Extra-Tree Classifier 15

2.2.2 K-means Clustering 15

2.2.3 Hidden Markov Model 17

2.2.4 Neural Network - Auto-Encoder 18

2.2.5 Gaussian Distribution Model 20

2.3 User Authentication 21
Chapter 3 24
3 Big Data Anomaly Detection Classification 24
3.1 Velocity - Time Complexity Classification 24
3.2 Variety - Data Nature Classification 25
3.3 Volume - Data Feature Classification 26
3.4 Comparison Study 27
3.5 Summary 30
Chapter 4 31
4 Proposed Anomaly Detection System 31
4.1 General Architecture 32
4.2 Anomaly Detection - Machine Learning Models 34
4.2.1 K-means Clustering, HMM, and Auto-encoder Models 34
4.2.2 Auto-Encoder-K-means and Auto-Encoder-HMM Models 35
4.2.3 Combination Model (Auto-encoder, K-means, and HMM) 36
4.2.4 Gaussian Distribution Model 37
4.3 Programming, Libraries and Evaluation Methods 38
4.3.1 Program Libraries 38
4.3.2 Common Evaluation Methods 40
4.3.3 Sequential Accuracy Algorithm (SAA) 42
4.3.4 Parameters Tuning 44
4.4 Anomaly Detection Results 47
4.4.1 Experiment 1 - Credit Card Dataset 48
4.4.2 Experiment 2 - Synthetic Dataset from a Financial Payment System 56
4.4.3 Experiment 3 - German Credit Risk Dataset 64
4.4.4 Experiment 4 - Server Computers Dataset 75

4.4.5 Experiment 5 - High Dimensional Server Computers Dataset 83
4.4.6 Experiment 6 - Transmission History Dataset 91
4.4.7 Experiment 7 - Porto Seguro's Safe Driver Prediction Dataset..... 99
4.4.8 Experiment 8 – Santander Customer Transaction Dataset 107
4.4.9 Experiment 9 - Prudential Life Insurance Assessment Dataset 115
4.4.10 Results Summary and Experiments Conclusion 125
Chapter 5 127
5 User Authentication 127
5.1 "Something you do"-Based Authentication 127
5.2 User Profile 129
5.3 Creating an Individual User Profiles 134
5.4 Challenging Questions 141
Chapter 6 144
6 Conclusion and Future Works 144
6.1 Conclusion 144
6.2 Future Works 146
References 147
Appendices 1
Curriculum Vitae 1

List of Tables

Table 3.1: Anomaly Detection Factors with Big Data Characteristics 24

Table 3.2: SVM and Neural Network Comparison Table 28

Table 3.3: K-means, HMM, Auto-encoder, and Gaussian Distribution Comparison Table .. 29

Table 4.1: Data Splitting in Anomaly Detection System 33

Table 4.2: Used Python Libraries and Description 38

Table 4.3: Confusion Matrix Table 40

Table 4.4: Tuning Parameters in Python 45

Table 4.5: Dataset 1 Description 48

Table 4.6: Results for Dataset 1 based on Four Assumptions 51

Table 4.7: K-means Results for Dataset 1 52

Table 4.8: Parameters Ranges 53

Table 4.9: HMM Results for Dataset 1 53

Table 4.10: Auto-Encoder Model Results 54

Table 4.11: Results of Four Models 55

Table 4.12: Dataset 2 Description 57

Table 4.13: Results for Dataset 2 based on Four Assumptions 59

Table 4.14: K-means Results for Dataset 2 60

Table 4.15: HMM Results for Dataset 2 61

Table 4.16: Auto-Encoder Model Results for Dataset 2 62

Table 4.17: Results of Four Models for Dataset 2 63

Table 4.18: Dataset 3 Description 66

Table 4.19: Results for Dataset 3 based on Four Assumptions 69

Table 4.20: K-means Results for Dataset 3 70

Table 4.21: HMM Results for Dataset 3 71

Table 4.22: Auto-Encoder Model Results for Dataset 3 73

Table 4.23: Results of Four Models for Dataset 3 74

Table 4.24: Dataset 4 Description 75

Table 4.25: Results for Dataset 4 based on Four Assumptions 78

Table 4.26: K-means Results for Dataset 4 79

Table 4.27: HMM Results for Dataset 4 80

Table 4.28: Auto-Encoder Model Results for Dataset 4 81

Table 4.29: Results of Four Models for Dataset 4 82

Table 4.30: Dataset 5 Description 83

Table 4.31: Results for Dataset 5 based on Four Assumptions 86

Table 4.32: K-means Results for Dataset 5 87

Table 4.33: HMM Results for Dataset 5 88

Table 4.34: Auto-Encoder Model Results for Dataset 5 89

Table 4.35: Results of Four Models for Dataset 5 90

Table 4.36: Dataset 6 Description 91

Table 4.37: Results for Dataset 6 based on Four Assumptions 94

Table 4.38: K-means Results for Dataset 6 95

Table 4.39: HMM Results for Dataset 6 96

Table 4.40: Auto-Encoder Model Results for Dataset 6 97

Table 4.41: Results of Four Models for Dataset 6 98

Table 4.42: Dataset 7 Description 99

Table 4.43: Results for Dataset 7 based on Four Assumptions 102

Table 4.44: K-means Results for Dataset 7 103

Table 4.45: HMM Results for Dataset 7 104

Table 4.46: Auto-Encoder Model Results for Dataset 7 105

Table 4.47: Results of Four Models for Dataset 7 106

Table 4.48: Dataset 9 Description 107

Table 4.49: Results for Dataset 8 based on Four Assumptions 110

Table 4.50: K-means Results for Dataset 8 111

Table 4.51: HMM Results for Dataset 8 112

Table 4.52: Auto-Encoder Model Results for Dataset 8 113

Table 4.53: Results of Four Models for Dataset 8 114

Table 4.54: Data Features Description for Dataset 9 116

Table 4.55: Dataset 9 Description 117

Table 4.56: Results for Dataset 9 based on Four Assumptions 120

Table 4.57: K-means Results for Dataset 9 121

Table 4.58: HMM Results for Dataset 9 122

Table 4.59: Auto-Encoder Model Results for Dataset 9 123

Table 4.60: Results of Four Models for Dataset 9 124

Table 4.61: Best Model per Experiment 126

Table 5.1: User Profile Specification Features 130

Table 5.2: User Profiles Sample from Experiment 1 131

Table 5.3: User Profiles Sample from Experiment 2 131

Table 5.4: User Profiles Sample from Experiment 3 132

Table 5.5: User Profiles Sample from Experiment 6 132

Table 5.6: User Profiles Sample from Experiment 7 133

Table 5.7: User Profiles Sample from Experiment 9 133

Table 5.8: User Profiles Comparison Table 135

Table 5.9: a Sample of User Profile 140

List of Figures

Figure 1.1: The main components of Ouda's user authentication framework [1] 2

Figure 1.2: Nine Experiments, Fields and Sizes 7

Figure 2.1: Anomaly Detection Categories 9

Figure 2.2: Anomaly Detection Techniques Types and Examples 10

Figure 2.3: Anomaly Detection Diagram 12

Figure 2.4: K-means Clusters 16

Figure 2.5: HMM Diagram 17

Figure 2.6: Simple Artificial Neural Network 19

Figure 2.7: Auto-Encoder neural network Model 20

Figure 2.8: User Authentication Techniques 22

Figure 3.1: Velocity - Time Complexity Classification 24

Figure 3.2: Big Data Sources and Types 25

Figure 3.3: Variety – Data Nature Classification 26

Figure 3.4: Volume - Data Feature Classification 27

Figure 4.1: Used Machine Learning Techniques and their purposes 31

Figure 4.2: Anomaly Detection Proposed Architecture 32

Figure 4.3: K-means Clustering, HMM, and Auto-encoder Models 35

Figure 4.4: Auto-Encoder-K-means and Auto-Encoder-HMM Models 36

Figure 4.5: Auto-Encoder, K-means, and HMM Model 37

Figure 4.6: Gaussian Distribution Model..... 38

Figure 4.7: Features Histogram for Dataset 1..... 49

Figure 4.8: Feature Importance for Dataset 1..... 50

Figure 4.9: The Best Results in Experiment 1..... 56

Figure 4.10: Features Histogram for Dataset 2..... 57

Figure 4.11: Feature Importance for Dataset 2..... 58

Figure 4.12: The Best Results in Experiment 2..... 64

Figure 4.13: Features Histogram for Dataset 3..... 67

Figure 4.14: Feature Importance for Dataset 3..... 68

Figure 4.15: The Best Results in Experiment 3..... 75

Figure 4.16: Features Histogram for Dataset 4..... 76

Figure 4.17: Feature Importance for Dataset 4..... 77

Figure 4.18: The Best Results in Experiment 4..... 83

Figure 4.19: Features Histogram for Dataset 5..... 84

Figure 4.20: Feature Importance for Dataset 5..... 85

Figure 4.21: The Best Results in Experiment 5..... 91

Figure 4.22: Features Histogram for Dataset 6..... 92

Figure 4.23: Feature Importance for Dataset 6..... 93

Figure 4.24: The Best Results in Experiment 6..... 99

Figure 4.25: Features Histogram for Dataset 7..... 100

Figure 4.26: Feature Importance for Dataset 7 101

Figure 4.27: The Best Results in Experiment 7 107

Figure 4.28: Features Histogram for Dataset 8 108

Figure 4.29: Feature Importance for Dataset 8 109

Figure 4.30: The Best Results in Experiment 8 115

Figure 4.31: Features Histogram for Dataset 9 118

Figure 4.32: Feature Importance for Dataset 9 119

Figure 4.33: The Best Results in Experiment 9 125

Figure 4.34: TNR and TPR for the highest result in every Experiment. 126

Figure 5.1: Security Questions Types and Examples 129

Figure 5.2:User Behaviour Modeling Diagram. 136

Figure 5.3: Anomaly Detection Model. 137

Figure 5.4: User Behavuior Modeling Diagram. 137

List of Appendices

Appendix A: All the results for all data in assumption 1 (without normalization or dimensional reduction)..... 1

Appendix B: All the results for all data in assumption 2 (with normalization only)..... 3

Appendix C: All the results for all data in assumption 3 (with dimensional reduction only) .. 5

Appendix D: All the results for all data in assumption 4 (with both normalization dimensional reduction only) 7

Appendix E: PCA comparison based on features for experiment 1. 9

Appendix F: PCA comparison based on features for experiment 2..... 10

Appendix G: PCA comparison based on features for experiment 3. 10

Appendix H: PCA comparison based on features for experiment 4. 11

Appendix I: PCA comparison based on features for experiment 6. 12

Appendix J: PCA comparison based on features for experiment 7. 12

Appendix K: PCA comparison based on features for experiment 8. 14

Appendix L: PCA comparison based on features for experiment 9. 15

Appendix M: All results in K-means Model for Experiment 1. 15

Appendix N: All results in HMM Model for Experiment 1. 18

Appendix O: All results in Auto-Encoder Model for Experiment 1. 19

Appendix P: All results in K-means Model for Experiment 2..... 22

Appendix Q: All results in HMM Model for Experiment 2. 24

Appendix R: All results in Auto-Encoder Model for Experiment 2. 26

Appendix S: All results in K-means Model for Experiment 3..... 27

Appendix T: All results in HMM Model for Experiment 3..... 30

Appendix U: All results in Auto-Encoder Model for Experiment 3..... 31

Appendix V: All results in K-means Model for Experiment 4..... 34

Appendix W: All results in HMM Model for Experiment 4. 36

Appendix X: All results in Auto-Encoder Model for Experiment 4..... 38

Appendix Y: All results in K-means Model for Experiment 5..... 39

Appendix Z: All results in HMM Model for Experiment 5..... 42

Appendix AA: All results in Auto-Encoder Model for Experiment 5..... 43

Appendix BB: All results in K-means Model for Experiment 6. 46

Appendix CC: All results in HMM Model for Experiment 6..... 49

Appendix DD: All results in Auto-Encoder Model for Experiment 6..... 50

Appendix EE: All results in K-means Model for Experiment 7..... 53

Appendix FF: All results in HMM Model for Experiment 7..... 55

Appendix GG: All results in Auto-Encoder Model for Experiment 7..... 57

Appendix HH: All results in K-means Model for Experiment 8..... 59

Appendix II: All results in HMM Model for Experiment 8..... 62

Appendix JJ: All results in Auto-Encoder Model for Experiment 8. 63

Appendix KK: All results in K-means Model for Experiment 9..... 66

Appendix LL: All results in HMM Model for Experiment 9. 68

Appendix MM: All results in Auto-Encoder Model for Experiment 9. 70

List of Abbreviations

AD	Anomaly Detection
NN	Neural Network
HMM	Hidden Markova Model
GD	Gaussian Distribution
DR	Detection Rate
TPR	True Positive Rate
TNR	True Negative Rate
DSA	Data Security-based Analytics
BDA	Big Data-driven authentication tool
JitHDA	Just-in-time human dynamics-based authentication engine
ML	Machine Learning
SVM	Support vector Machine
OCSVM	One Class Support Vector Machine
IDS	Intrusion Detection Systems
ANN	Artificial Neural Network
Prec	Precision
Rec	Recall
MLP	Multi-Layer Perceptron
RNN	Recurrent Neural Network

CNN	Convolutional Neural Network
AUC	Area Under Curve
LSTM	Long short-term memory
PCA	Principal Component Analysis
Roc	Receiver Operating Characteristic
Sqrt	Square Root
RMSE	Root Mean Square Error
Tol	Tolerance
SAA	Sequential Accuracy Algorithm
SNA	Social Network Analysis

Chapter 1

1 Introduction

In preventing and detecting unauthorized use of computer systems, user authentication is the first line-of-defense against cyber-attacks. RFC 2828 defines user authentication as the process of verifying an identity claimed by or for a system entity [1]. An authentication process consists of two steps: (1) presenting an identifier to the security system, and (2) presenting or generating authentication information that corroborates the binding between the user and the identifier. There are many user authentication methods that are implemented and used to provide secure user authentication. These methods can be classified under three main authentication categories. (i) "Something-you-know", examples include a password, a PIN number (ii) "Something-you-have", examples include cryptographic key generators and smart cards. (iii) "Something-you-are", examples include the recognition of users' fingerprint, iris, and face, known to be static biometric measures. Each of these methods has its own security advantages and pitfalls.

Ouda, [1] has developed a new framework to describe the rise of new generation user authentication systems. The framework is recommending the leverages of Big Data analytics and relying on a "something you do"-based verification process. Figure 1.1 shows the main component of this framework. The framework provided three main components that indicate the perspectives for the researchers to approach the development of strong user authentication systems. These components are: (1) Data Security-based Analytics (DSA) that describe ways to leverage Big Data analytics to have valuable insight of the users' data with the appropriate depth needed to deliver up-to-date representation of the user behaviour, (2) Big Data-driven Authentication tools (BDA), to analyze the captured user behaviour and focus on the sudden changes of the user's actions, along with the real-time uniquely identifiable information to build accurate patterns of the users' actions in the form of user security profile, and (3) Just-in-time human dynamics based authentication engine (JitHDA) that utilizes these profiles to generate a real-time (i.e., just in time) set of challenging questions. These questions should cover the unique actions that explicitly represent an instantaneous specific user's behaviour.

This thesis proposes a novel implementation model for Ouda's authentication framework. This model utilizes the Machine Learning-based Anomaly Detection technique to develop the security potential user profiles by which a structural database of challenging questions is constructed.

The following sections discuss the motivation for this work and thesis objectives. The methodology and the thesis contributions are addressed. Lastly, this chapter explains the thesis outline.

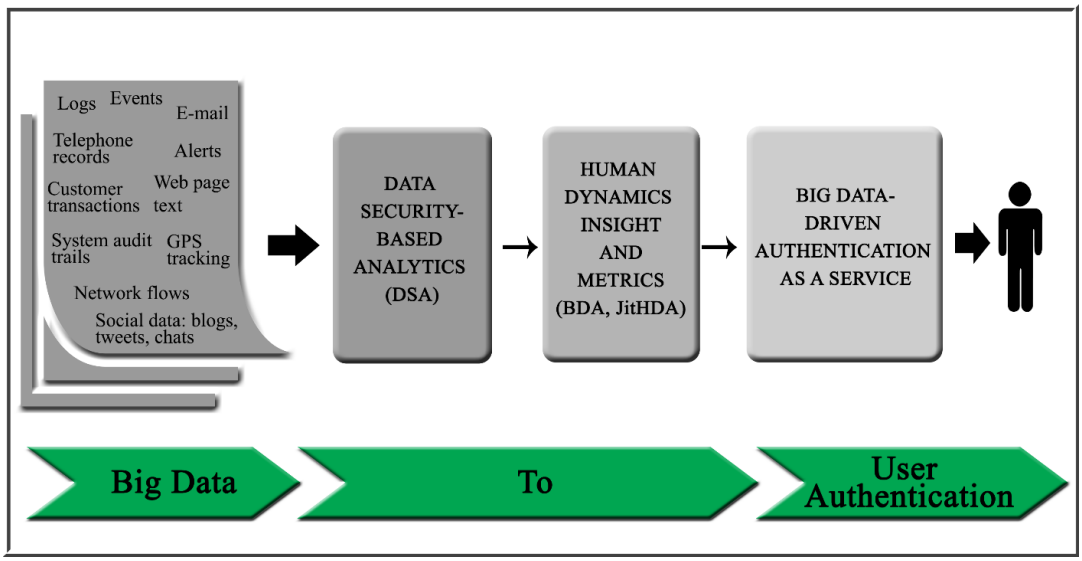

Figure 1.1: The main components of Ouda's user authentication framework [1]

1.1 Research Motivation

People spend a significant amount of time, in their daily routine, interacting with social network applications such as Twitter and Facebook. Every time people use credit cards, their purchase data is not only being tracked but also the products that are being sold to which group of customers are stored. People and companies are using cloud-based email services such as those services provided by Yahoo and Google. This is because they offer compelling functionalities and assign huge amounts of user repositories. These email providers are using algorithms to scan the email content for keywords aiming to offer some

advertisements toward user interests. For instance, a user may start getting links for hotel reservations just after receiving a confirmation email about an airline booking.

Having said the above, we believe that many aspects of users' traits would be digitally captured in real-time or accumulated for future data analysis. This has turned our attention to the fact that, with proper analysis of this data an accurate detection of people's behaviours can be made and hence their identification factors can be verified, especially when the results of this analysis are fed into user authentication methods. However, the continuous flow of high-volume data requires sophisticated data analysis techniques to be able to examine huge amounts of behavioural evidence so that user traits can be modeled. In addition, these techniques should have the ability to distinguish between normal and abnormal actions of users, so that security potential data can be captured.

In this regard, we are in the favor of enhancing the anomaly detection techniques to be utilized for users' trait analysis in an attempt that the detected information will fulfill the needs for the user's identity verification.

1.2 Research Objectives

The main goal of this research is to build a users' behaviour analyzer engine to automatically and accurately detect a range of abnormal actions among high-volume, fast, and mutable streams of users' data. The result of these detections should be enough to structure and develop security user profiles. These profiles provide an image of sensitive information about the users by which a strong real-time user authentication model can be designed. In other word, the main goal of this research is to design and implement accurate and complete models for the DSA, BDA, and JitHDA components within Ouda's authentication framework described above. It worth mentioning that, this work has been build based on the assumption that, all data source is free from any fraud transactions.

The following are the research objectives that support the above goal.

1. Investigate anomaly detection techniques and recent innovative research done in this area. Also, study Big Data characteristics especially for anomaly detection techniques

and then chose the most effective characteristics to build a novel study for anomaly detection in Big Data applications.

2. Based on the previous objective (Study for anomaly detection techniques in Big Data), develop an anomaly detection model that is suitable for Ouda's user authentication framework with choosing the best evaluation method.

3. Create a prototype for user authentication systems using anomaly detection outcomes by generating a sample of user profiles.

1.3 Research Methodology

This section describes the methodologies that are applied in this research for each objective to design and implement the anomaly detection for user the authentication framework as follows:

Objective one is a novel study for anomaly detection techniques based on Big Data which can be completed by the following tasks:

- Explore all anomaly detection techniques including the recent research that is related to Big Data applications.

- Study the Big Data characteristics, sources, features, and applications and choose the most common V's related to anomaly detection problems.

- Extract three factors in anomaly detection techniques through the recent research that match or are related to the chosen Big Data characteristics.

- Identify and classify the collected anomaly detection techniques based on the factors – Big Data characteristics combination from the previous task.

- Create two comparative studies for the most common techniques in supervised and unsupervised learning for the recent research papers with specific factors for all chosen papers and some conditions to choose the papers.

Objective two is designing an anomaly detection model which can be completed by the following tasks:

- Choose the most commonly used unsupervised techniques based on Big Data anomaly detection classification and the comparison study provided.

- Apply most of the popular binary evaluation methods to choose the suitable one for our research case and develop two sequential accuracy algorithms to make sure the existing evaluation methods calculate the sequential accuracy.

- Apply the chosen unsupervised techniques from task one in this objective and tune them with several parameters on nine different experiments.

- Assume different models that are combined from the chosen techniques to get more analyzation and accuracy.

- Obtain the best model with the best accuracy for every experiment.

Objective Three is developing a user authentication prototype which can be completed by the following tasks:

- Choose and analyze the experiment results that are suitable for user profile generation using a specific criterion.

- Design and create user profiles for a sample of anomalous cases from the suitably chosen anomaly detection results for profile features that are compatible with the Ouda's user authentication framework.

- Provide a scenario for creating challenging questions based on the user profiles for user authentication recommending specific rules to match the high level of security.

- Validate the final challenging questions in the user authentication framework with strong examples from the user profiles.

1.4 Research Contribution

This thesis focuses on designing and implementing an anomaly detection technique suite for Ouda's user authentication framework. Initially, it offers a study on Big Data for anomaly detection techniques which has three classifications. These classifications are completed based on three Big Data characteristics that are related to the three factors in anomaly detection techniques; Volume with data features, Variety with the natural types of data, and Velocity with computational complexity. Each one of the classifications describe the common machine learning (ML) techniques that are used in recent research. These classifications helped me to choose the best model fit with the best problem. Two comparison studies (supervised and unsupervised techniques) over a number of recent research papers are presented for the chosen ML models with specific comparison factors and some research paper standards.

This thesis also proposes an anomaly detection (AD) model that contains a combination of several techniques that are suitable for Big Data applications. The AD models are combined with several machine learning techniques; K-means, Hidden Markov Model (HMM), Auto-Encoder NN, and Gaussian Distribution. In total, the applied models and techniques are seven; the four basic techniques and three combined as follows: 1) K-means with Auto-encoder NN, 2) HMM with Auto-encoder NN, and 3) K-means, HMM and Auto-encoder NN. These models are applied on nine different experiments and give good detection results. The experiments are applied to a variety of fields such as financial payment systems, insurance systems (health, auto, home), computer server monitoring systems, and network transmission systems. Figure 1.2 shows the nine experiments related to the fields and sizes. Most of the common evaluation methods are applied in this thesis. Confusion matrix, true positive rate (TPR), and true negative rate (TNR) are chosen for comparing the results because they match the research needs.

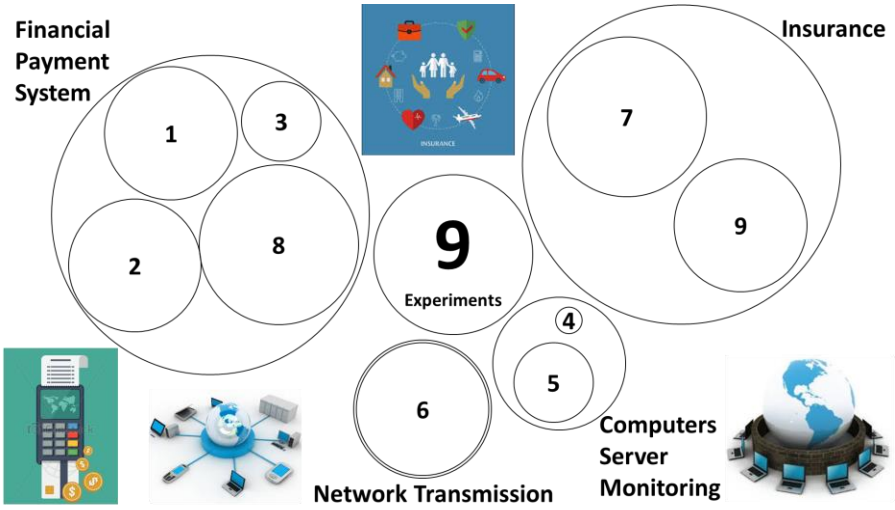

Figure 1.2: Nine Experiments, Fields and Sizes

Finally, this thesis proposes a scenario of generating security questions based on a desired anomaly detection model and user profiles. This scenario provides strong examples of challenging questions from a sample of user profiles that are created after anomaly detection analyzation has been done on Big Data.

The research contributions of this thesis have been published in several conference proceedings in the areas of information security and data analytics. Therefore, these contributions have been peer-reviewed by experts in the field.

1.5 Research Outline

The thesis structure is ordered as follows. Chapter 2 provides a literature review of anomaly detection techniques and background on the user authentication system as well as theoretical information of the most commonly used anomaly detection techniques. In Chapter 3 we present and discuss anomaly detection techniques in Big Data applications by providing three classifications for the commonly used anomaly detection techniques. An anomaly detection model is discussed in high detection accuracy as well as how this final model is combined and chosen with result discussions in Chapter 4. Chapter 5 discusses a scenario on how a challenging question would be created using anomaly detection results including how user profile generation is achieved. Chapter 6 concludes with the thesis and addresses the future work recommendations and directions.

Chapter 2

2 Literature Review and Background

This chapter presents a literature review of the current anomaly detection techniques on Big Data and the known classifications. It also presents an in-depth concept of anomaly detection and its mechanism in some applications as well as commonly used anomaly detection techniques. Finally, it overviews user authentication techniques in general and explains more details in the related knowledge-based applications.

2.1 Literature Review

The term “anomaly” is defined as something that deviates from what is standard, normal, or expected. In data science, a data anomaly is not far from this definition. However, the deviation from the standard or expected data might be due to errors in the data or due to correct data that is triggered by uncommon, but accurate actions. In both cases, the detection of these deviations is desirable whether to correct the errors (if any), or to gain better insight on data. Many anomaly detection techniques exist in academic literature, and share the same purpose, that is to differentiate between what is normal and abnormal.

There are three broad categories of anomaly detection that are classified based on the type of the datasets they are working on, i.e., whether the data is labeled or not. Supervised anomaly detection techniques detect anomalous data based on the available labeled data for both anomalous and normal labels. Unsupervised techniques detect anomalous data based on unlabeled data. Semi-supervised anomaly detection techniques assume that the labels exist only for normal data, while the anomalous data is detected [2].

Under these three categories, anomaly detection techniques can be further divided into six subcategories. Although there are many classifications in the literature, we will address the most common approach among researchers. Figure 2.1 illustrates this classification approach. Classification techniques build classifiers based on labeled training sets to distinguish between normal and abnormal test data and are most likely used as a specific type of the supervised techniques. Nearest neighbour techniques utilize the

similarity or distance between samples to detect the anomalous data. Clustering techniques group the data to detect the individual or group anomalies among normal group data. Spectral techniques embed the data into a smaller subspace to find the differences between normal and abnormal data. These three groups are mainly used to further classify both the semi-supervised techniques. Moreover, statistical and informational theories would be used to classify the unsupervised techniques. Statistical techniques assume high probability for normal data and low probability for anomalous data. Information theory techniques detect anomalous data through the irregular information content in the dataset. The reason behind this classification is highlighted by the following scenarios. Each scenario describes the applicable types and examples that would be used.

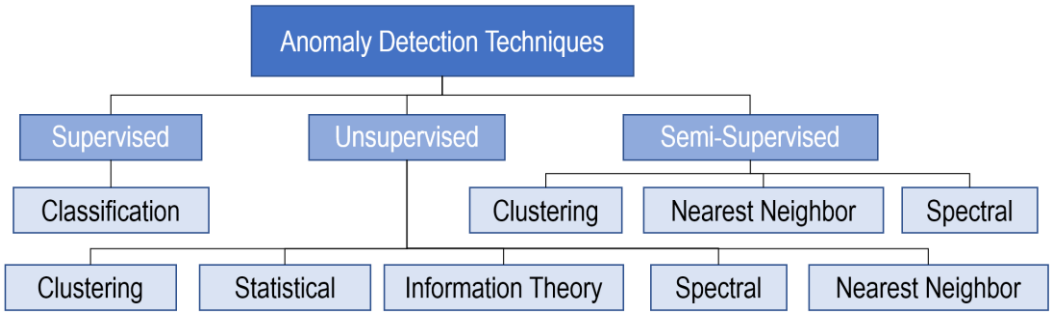

Figure 2.1: Anomaly Detection Categories

There are many popular classifiers that have been used in anomaly detection such as neural networks, support vector machine, Bayesian networks and rule [2] - [6]. In the nearest neighbour category, there are two types of techniques, namely, k^{th} nearest neighbour and density nearest neighbour. The former computes the anomaly score using the similarity between a data sample and its k^{th} nearest neighbour. However, the later computes anomaly score using the relative density of each data sample. Similarly, clustering techniques have three types based on three assumptions:

1) Anomalies do not belong to any cluster but normal data belongs to a cluster.
2) The closest data to a cluster centroid is normal data whereas the far data are anomalies.
3) The large clusters contain normal data yet anomalies exist in small clusters.

Statistical techniques can be divided into parametric and non-parametric types. The normal data is produced using a parametric distribution in parametric type such as Gaussian Model, Regression Model, and Mixture of Parametric Distributions. But a non-parametric type does not consider any parametric distribution such as histogram model and kernel function. Information theoretic techniques use several measures to analyze the information content using Kolomogorov complexity, entropy, and relative entropy. The spectral techniques use dimensional reduction techniques by employing Principal Component Analysis (PCA) and Compact Matrix Decomposition. Figure 2.2 summarizes the above scenarios including the examples and types of anomaly detection techniques.

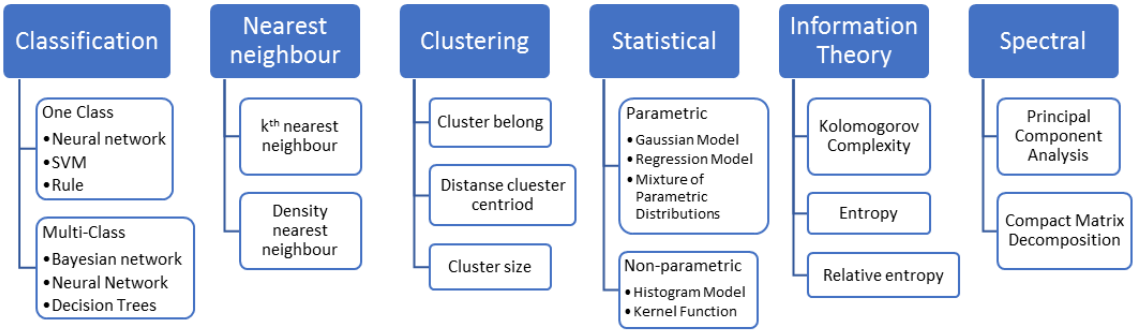

Figure 2.2: Anomaly Detection Techniques Types and Examples

Our aim in this work is to shed light on Big Data-enabled anomaly detection techniques. Researchers define Big Data as datasets that possess the characteristics of the 3Vs (Volume, Variety, and Velocity). Volume refers to the scale of the data. Variety refers to the heterogonous data presentations such as unstructured, semi-structured, and structured data. Velocity refers to the pace at which data is generated. When data becomes Big Data, the above classifications of anomaly detection needs to be reinvestigated (in a later chapter).

Chandola, et al. [2] discussed the anomaly detection techniques with several aspects. However, the authors do not include the characteristics of Big Data in their survey. Moreover, Rana, et al. [6] give guidelines for Big Data but it is specific to a data stream type. Other recent surveys study the characteristics of anomaly detection against some specific datasets. For instance, Wu [7] focuses on time series datasets which can ignore

other types of datasets. Also, Patil and Biswas [8] have an anomaly detection survey with only video datasets. While some surveys concentrate on some types of data, other research papers have an emphasis on a specific anomaly detection application. For example, Anand, et al. [9] Al-Musawi, et al. [10] have anomaly detection surveys on Border Gateway Protocol and online social networks respectively. Kaur and Singh [11], Fanaee-T and Gama [12], have anomaly detection surveys which include general information for most techniques without a real implementation.

This literature review includes many anomaly detection techniques that need to be explained. The next section will give the important background information for anomaly detection mechanisms and the techniques that will be used in this research.

2.2 Anomaly Detection Techniques

Generally, anomaly detection works with both supervised (detection of anomalous data based on the labeled data for both anomalous and normal labels) and unsupervised (detection of anomalous data based on unlabeled data) machine learning techniques. Furthermore, the reasons to prefer an unsupervised machine learning technique in anomaly detection systems, even if there is a labelled 0 for normal and 1 for anomaly data are:

- A small number of positive (anomalous) data
- A large number of negative (normal) data.
- The existence of many different types of anomalies, which makes it hard for an algorithm to learn, especially if positive data is small.
- And, in this work, the user authentication application requires to deal with unlabeled data (the labels will be used only for the evaluation part).

An anomaly detection approach is when an unlabeled training set is used to build a model P(x); where p is the type of model (probability, clustering, or hierarchy), and x is some data attributes (A.K.A. data features, or just features) of the unlabeled training set. Therefore, an anomaly detection model of x has been built, then new instances (a test set) should be analyzed. If p of x-test is less than some specific criteria such as the threshold probability value, then the model will flag it as an anomaly, as shown in Figure 2.3. The

model mechanism will be more transparent by explaining some existing anomaly detection applications.

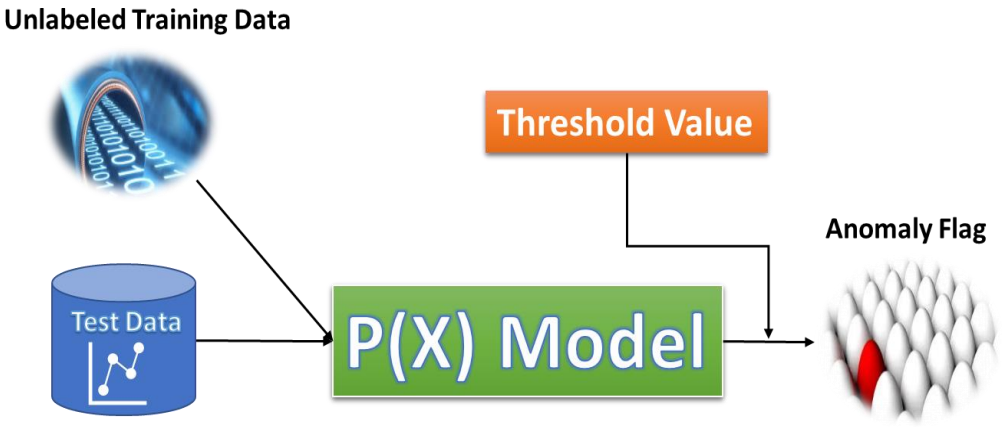

Figure 2.3: Anomaly Detection Diagram

The next section will explain in detail some examples of anomaly detection applications. Moreover, if a platform has many users, and each of these users takes different activities, the platform such as a website can compute different data features of users' activities.

Using these features, the model can be built to produce some results like, "what is the probability of different users behaving different ways?" and "what is the probability of features of a user's behaviour?" At this point, the user's activity features are known from the model results that is already built. An example of that could be "how often a particular user logs in or does transactions?"

Finally, the model can identify the strange user behaviour on the platform by checking the results under a threshold value. It can also create users' profiles for more analysers or request further verification from those users to guard the platform against strange or fraudulent behaviour. This system is used by many online platforms to detect not only stolen or fraudulent behaviour but also the abnormal behaviours for any further purposes.

Another anomaly detection application can be applied in the manufacturing process where unusual products could be found getting more reviews. These reviews can be used

to enhance future manufacturing. A third example of anomaly detection application is in monitoring computer systems in a data center that utilizes online and offline machine learning techniques to detect abnormal computer behaviours such as different amounts of memory use, different numbers of disc accesses, and different CPU loads. The machine learning techniques used in these applications are very widely different. However, there are several popular unsupervised machine learning techniques what will be explained in the next section and used in this research.

The purpose of this work is to build and create a unique knowledge-based authentication system that relies only on the abnormal actions of users to be the base of the challenging questions. This system utilizes the anomaly detection technique such that the answers of the challenging questions are known only by the legitimate user and easy to remember.

Anomaly detection techniques have been successfully used in Big Data applications, user profile-based systems, and unsupervised-based techniques. Recent research has increased in Big Data applications for anomaly detection system such as [13] – [17]. In [13], Gupta, et al. developed an advanced system with a highly accessible feature that is suitable for Hadoop clusters monitoring in real-time. In paper [14], Abu Sulayman and Ouda stated a unique vision for Big Data applications in anomaly detection techniques. This unique insight has a practical application using two machine learning techniques and three new classifications. Mehnaz and Bertino in this paper [15] suggested the anomaly detection approach which established strong user profiles by analyzing the timestamp data of users' files and the temporal characteristics using a multilevel temporal data structure. Henriques et al, presented machine learning techniques which have self-learning user profiles in IDS systems [16]. Research [17] proposed a technique that detected the trends of abnormal behaviour then alerts the administrator and the user in real-time. Three kinds of techniques; regression, unsupervised classification, and simple statistical techniques were tested. Sometimes, it is vital to have an anomaly detection system that is suitable in a specific Database.

Other recent research has explored user profile generation for anomaly detection in specific databases [18] – [22]. A database proposal is designed for anomaly detection to develop the accuracy of database anomaly detection and to generate the users' profiles accurately in [18]. A technique is proposed to find the anomalous data in database using a classification machine learning technique by Ramachandran et al. in [19]. Pannell and Ashman proposed an IDS system for a host-based behaviour that utilized user profiles in anomaly detection to characterize every behaviour by combining the results of multiple features to develop detection performance [20]. A software prototype is improved by Corney et al., which recognized anomalous data based on behaviour patterns, then alarms administrators when such data are recognized [21]. The research paper [22] introduced a novel user profiling mechanism which covered all accessible resources and relevant characteristics upon on the cybersecurity perspective. The proposed technique contained seven profiling principles to collect user information and more than 270 characteristics to generate the user security profile. Many machine learning techniques are suitable for user profile AD systems, although, clustering-based techniques, HMM's, and Auto-encoder neural networks are more commonly used in recent AD research.

K-means clustering based technique has been increased in recent research in AD systems. Jeyauthmigha and Suganthe designed a network anomaly detection frame with three clustering techniques in two stages: training and detection. The stages used three algorithms computed one after another. One of the algorithms is K-means clustering [23]. Ahmed proposed a hybrid technique for the anomaly detection framework. The hybrid technique has two algorithms: one is clustering the input network traffic dataset to create a collective anomaly, and one is re-clustering [24].

Iyer, et al. [25] presented fraud detection using a Hidden Markov Model, which is trained with the normal user behaviour and tested for both normal and fraud user behaviour. Also, they compared HMM with other methods to prove that HMM is the more preferred method. Zhu, et al. introduced a framework for anomaly detection using the Hidden Markov Model and Support Vector Machine to detect the abnormal events. They deployed the method on an IDS system to evaluate results [26]. Rahmani and Almasganj utilized auto-encoder and HMM to detect three different types of visual features inside a lip-reading

task [27]. Wang, et al. described the entire process of fraud detection using the Hidden Markov model and K-means algorithm. The model is trained using the normal user behaviour account to detect not accepted behaviour by considering the high probability as fraudulent [28].

Our approach compares three machine learning techniques; K-means clustering-based technique, HMM model, and Auto-encoder neural networks to detect anomalies in high accuracy as part of a user authentication framework. These three techniques have the different internal structure to discover the anomalous data. The understanding of internal structure improves the implementation results. Though, the internal structure of these techniques is explained briefly in the following subsections to simplify the resulting discussion.

2.2.1 Extra-Tree Classifier

Extra Tree (**ext**remely **ra**ndomized **trees**) classifiers are an ensemble learning method fundamentally based on decision trees. It randomizes certain decisions and subsets of data to minimize over-learning from the data. It builds multiple trees and splits nodes using random subsets of features. More variation in the ensemble will introduce how we can build trees [29]. Each decision base will be built with the following standards:

- All the data available in the training set is used to build each stump.
- Any node is performed using the best split which is determined by searching in a subset of randomly selected features. The split of each selected feature is chosen at random.
- The maximum depth of the decision base is one.

2.2.2 K-means Clustering

K-means clustering is one of the unsupervised anomaly detection techniques that proves its' high accuracy results in this domain. The main idea of the K-means clustering technique is to initialize several centroids Ks (as shown in Figure 2.4) based on randomly generated points within the data domain. Then, it will calculate the distance between every instance and the nearest centroid to this instance. After that, a step will occur to update the

centroid's positions based on the distance calculation. At the end, every data sample n should belong to the nearest cluster.

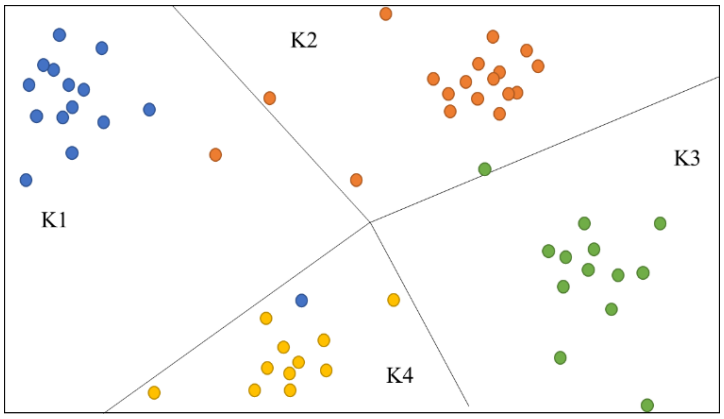

Figure 2.4: K-means Clusters

Clustering is a process of classifying data observations into different classes. Each cluster has a centroid. The observations in one cluster have great similarity, but observations between different clusters have less similarity. Suppose $X = \{x_1, x_2, x_3, \cdots, x_n\}$ is a dataset in a given space. The data observation is classified by n numbers of clusters where C $(1 < C < n)$ clusters based on their similarity. The cluster centroids are:

$$\boldsymbol{C_r} = \frac{\mathbf{1}}{\boldsymbol{n_r}} \sum_{\boldsymbol{i=1}}^{\boldsymbol{n_r}} \boldsymbol{X_i^{(r)}} \qquad (2.1)$$

The objective function of clusters is:

$$\boldsymbol{min} \sum_{\boldsymbol{r=1}}^{\boldsymbol{C}} \sum_{\boldsymbol{j=1}}^{\boldsymbol{n_i}} \left| \boldsymbol{X_j^{(i)}} - \boldsymbol{C_i} \right|^{\mathbf{2}} \qquad (2.2)$$

Where $i = 1,2,3, \cdots, n$; n_r is the number of data observations in cluster r; represents that data observation (X_i) belongs to cluster r ; $r = 1,2, \cdots, C$; C $(1 < C < n)$ represents the number of cluster centroids; and n is the total number of data observations in the dataset [30], [31]. Finally, the algorithm can be summarized in five steps:

1) Cluster centroids initialization.

2) Assign data observations to clusters

3) Calculate the similarity between observations and centroid.

4) Update the cluster centroids positions

5) Repeat steps 2, 3, and 4 until no movement for centroids.

2.2.3 Hidden Markov Model

The Hidden Markov Model (HMM) has two hierarchy levels, which makes a multiple embedded stochastic process. HMMs can be used to analyze much more complicated stochastic processes as compared to a traditional Markov model. HMMs contain a set of transition probability matrices related to a finite set of states. The state outcome or instance is produced using an accompanying probability distribution. It is only the outcome and not the state that is visible to an external observer. HMMs have many typical applications in various areas such as speech recognition, bioinformatics, and genomics. Three main components can characterize an HMM as the following list and Figure 2.5 explain:

- X is the number of states in the model.
- Y is the number of distinct observation symbols per state. The observation symbols correspond to the physical output of the system being modeled.
- The green and black lines in Figure 7 present the state transition and the output probabilities matrix, respectively.

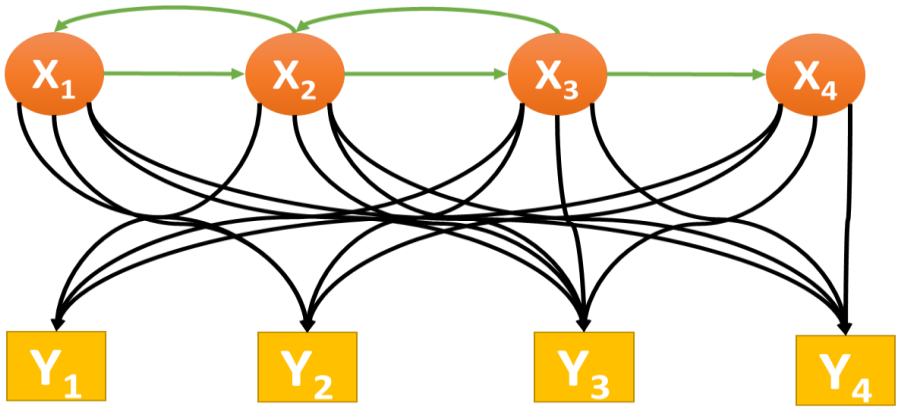

Figure 2.5: HMM Diagram

The HMM is a doubly stochastic model, expanded from the basic Markov model. A Markov chain contains a set of states, $S = \{s_1, s_2, s_3, \ldots., s_r\}$. The process starts in one of these states and moves successively from one state to another. The probability of moving from one state to another does not depend on which states the chain was in before the current state.

HMM is an underlying stochastic process that is not observable but can only be observed through another set of stochastic processes that produce the sequence of observed symbols. An HMM is notated as $\lambda = (A, B, \pi)$, where, A is the state transition probability matrix, B is the observation symbol probability matrix, and π is initial state probability vector.

There are three key problems for HMM when given the observation sequence $O = \{O_1, O_2, O_3, \ldots. O_T\}$ and the HMM $\lambda = (A, B, \pi)$:

- How to work out the probability $\Pr(O|\lambda)$.
- How to choose a state sequence $I = \{i_1, i_2, i_3, \cdots, i_r\}$
- How to adjust the model $\lambda = (A, B, \pi)$ parameters to maximize $\Pr(O|\lambda)$.

HMM is a powerful model for anomaly detection. We can use HMM to build a model of normal behaviour where the HMM's states represent some unobservable conditions of the system [32]. The HMM based anomaly detection method takes the following steps:

1) Train HMM based on normal observations.
2) Calculate the system state of the normal behaviour.
3) Calculate the system state of the new data behaviour.
4) Detect anomalies.

2.2.4 Neural Network - Auto-Encoder

Artificial neural network (ANN) is one of the most common network architectures. Basically, a simple artificial neural network only includes one or two hidden layers in addition to the input layer and output layer, from which is also a processing component similar to the hidden layers as shown in Figure 2.6. Furthermore, the input layer receives

the dataset. Then, the hidden layer can be one or more layers based on the problem complexity and the neural network type. Finally, the output layer will generate the result of this technique. The number of neurons for each layer depends on the data size and network type. All the neurons - except the output one - are connected to the neurons in the next layer with weights values. A neural network has several techniques that are frequently used in anomaly detection classifications due to their capability to classify the classes of datasets and their high accuracy in noisy data. These techniques are applicable to one class and multiclass problems. Feed forward neural, Auto-encoder neural, Recurrent neural, and Convolutional neural networks are the most popular neural networks that are used for anomaly detection techniques.

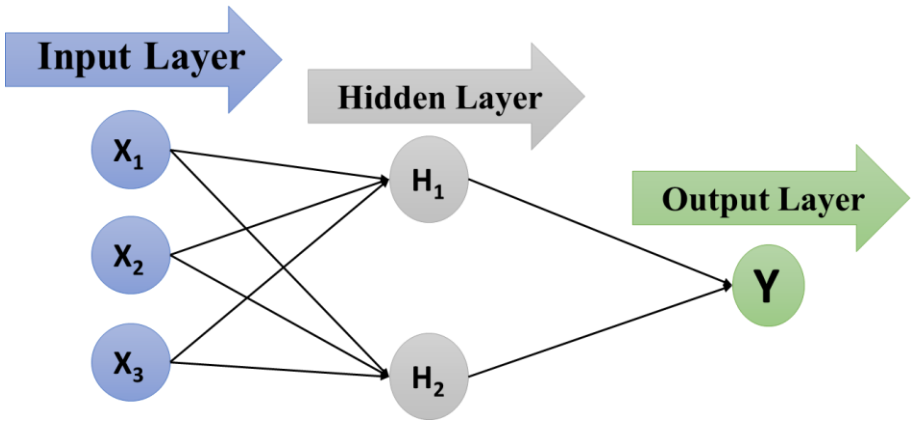

Figure 2.6: Simple Artificial Neural Network

Auto-encoders are a form of neural networks that attempt to learn an approximation of the identity function and reproduce the input to the output format. Accordingly, auto-encoders do not require any label or output to be trained or learn how to reconstruct the input. A simple auto-encoder can be formed from an input layer, one hidden layer and an output layer. The hidden layer usually has a smaller dimension than the input layer in order to learn the latent space representation of the input. The output layer usually has the same dimensions of the input layer since it is trying to predict it. Figure 2.7 shows a basic diagrammatic representation of an auto-encoder.

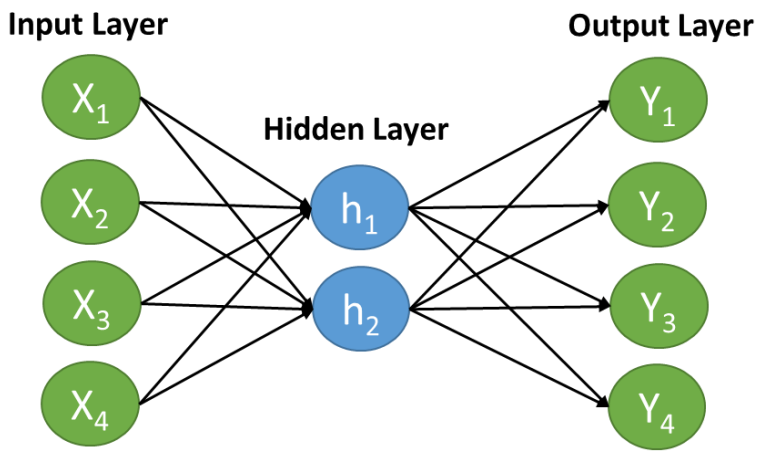

Figure 2.7: Auto-Encoder neural network Model

An auto-encoder includes two parts: encoder and decoder. The encoder aims to compress input data into a low-dimensional representation, and the decoder reconstructs input data based on the low-dimension representation generated by the encoder. Furthermore, an auto-encoder can encode a representation of an input layer into a hidden layer and then decode it into an output layer [33].

The auto-encoder based anomaly detection method takes the following steps:

1) Encoding the input data.
2) Reconstruct the data through the decoding.
3) Calculate the reconstruction error.
4) Use a threshold value for the reconstruction error to assign anomalies data.

2.2.5 Gaussian Distribution Model

To perform anomaly detection through Gaussian distribution, there is a need for data distribution. Given a training set $\{x^{(1)}, \cdots, x^{(m)}\}$; where $x^{(i)} \in R^n$ the Gaussian distribution should be estimated for each of the features. For each feature $i = 1, \cdots, n$, the parameters μ_i and σ_i^2 that fit the data in the i-th dimension should be found for each example.

The Gaussian distribution is given by equation 3:

$$\boldsymbol{\rho}(\boldsymbol{x};\boldsymbol{\mu},\boldsymbol{\sigma^2}) = \frac{\mathbf{1}}{\sqrt{\mathbf{2\pi\sigma^2}}}\boldsymbol{e}^{-\frac{(x-\mu)^2}{2\sigma^2}} \qquad (2.3)$$

Where μ is the mean and σ^2 controls the variance. Gaussian parameters which are (μ_i σ_i^2) of the i-th feature will be estimated using equation 4 for the mean and equation 5 for the variance.

$$\boldsymbol{\mu_i} = \frac{\mathbf{1}}{\boldsymbol{m}}\sum_{\boldsymbol{j=1}}^{\boldsymbol{m}} \boldsymbol{x_i^{(j)}} \qquad (2.4)$$

$$\boldsymbol{\sigma_i^2} = \frac{\mathbf{1}}{\boldsymbol{m}}\sum_{\boldsymbol{j=1}}^{\boldsymbol{m}} (\boldsymbol{x_i^{(j)}} - \mu_{\boldsymbol{i}})^{\mathbf{2}} \qquad (2.5)$$

The first function is to take the input data and output an n-dimension vector mu that holds the mean of all the n features and another n-dimension that holds the variances of all the features. After calculating the parameters, we need to select a threshold.

One way to determine which examples are anomalies is to select a threshold based on an F1 score on a cross validation set. The F1 score is computed using precision (*prec*) and recall (*rec*) using equation 6, 7, and 8:

$$\boldsymbol{F_1} = (\mathbf{2}\cdot\boldsymbol{prec}\cdot\boldsymbol{rec})/(\boldsymbol{prec}+\boldsymbol{rec}) \qquad (2.6)$$

$$\boldsymbol{prec} = \mathbf{tp}/(\mathbf{tp}+\mathbf{fp}) \qquad (2.7)$$

$$\boldsymbol{rec} = \mathbf{tp}/(\mathbf{tp}+\mathbf{fn}) \qquad (2.8)$$

Where tp is the number of true positives, fp is the number of false positives, and fn is the number of false negatives.

2.3 User Authentication

Recently, user authentication has become the most popular topic in information security research environments. The definition of user authentication is stated as the process of verifying an identity claimed by a user for a system entity. An authentication

challenge is a method used to distinguish between true or false authentication requests. User authentication has a variety of techniques that can identify the valid users in protected resources as it is shown in Figure 2.8. User authentication can be broadly classified into four groups based on something the user "is", "knows", "has", and "does". Usually, body parts are used in "something the user is" which are called biometric technology such as a fingerprint. Mostly, "Something the user has" uses a physical (non–body parts) thing to authenticate the user, for example, cards, keys, and so on. "Something the user knows" uses the user's knowledge such as an ID number, or Password. "Something the user does" is a new user authentication process that has been researched in recent years. This uses the user's activities such as Knowledge-based authentication (KBA) [34].

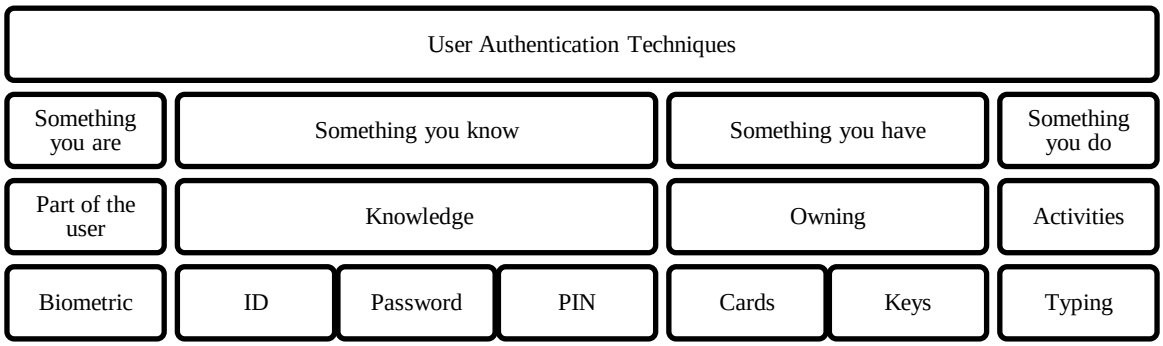

Figure 2.8: User Authentication Techniques

KBA is an authentication system in which the user should answer a set of challenging questions (or at least one) to be authorized. Generally, the challenging questions have two major categories; static and dynamic [35]. The static questions are the most commonly used, but it is considered weak authentication. One common application for a static security questions is "Fallback Authentication" that is a backup for authentication techniques in the lost cases. Moreover, fallback authentication is usually used when people lose their authentication access due to changes or forgetting the authentication requirements such as forgetting a password or username. Fallback authentication identifies the user through personal information and allows the authenticated user to re-access their resources [36]. However, this static question is a vulnerable way to

ask in Fallback Authentication because the answers can be found easily in many sources, especially in social media [37].

The second type of challenging questions have more invulnerability than the first type due to the dynamic way of asking the questions. These dynamic questions are generated using credit or a public user's information, which makes it sometimes easy to find, especially in social media apps [35]. The stronger way to produce a secure dynamic question achieves a more secure system against any fraudulent or abnormal activities [39].

As a result, unique dynamic security questions should be investigated with several features; a set of challenging questions based on abnormal user activities using short term history and is not repeated. This new way of asking the dynamic security questions can be generated based on studying the abnormal activities of the user behaviour utilizing anomaly detection.

Chapter 3

3 Big Data Anomaly Detection Classification

The literature review in chapter 2 shows that the most common classifications in anomaly detection techniques have a lack of Big Data insights. Our main contribution in this chapter is to shed light on Big Data anomaly detection techniques. In this chapter, three classifications of anomaly detection techniques in Big Data will be provided based on the Big Data definition in chapter 2. Three specific factors in anomaly detection techniques will be considered for the classifications with the related three big data characteristics. The factors combine with the characteristics as shown in Table 3.1.

Table 3.1: Anomaly Detection Factors with Big Data Characteristics

Anomaly Detection Factors	Time Complexity	The Nature of the Data	The Data Features
Big Data Characteristics	Velocity	Variety	Volume

3.1 Velocity - Time Complexity Classification

Anomaly detection can act as two major categories based on computational complexity, because the velocity of big data will affect the algorithm's time, including all the previous categories as shown in Figure 3.1. Linear computational complexity is a lower time complexity for the techniques. On the other hand, quadratic computational complexity is a higher time complexity. In addition, new types of applications for anomaly detection have been recently raised.

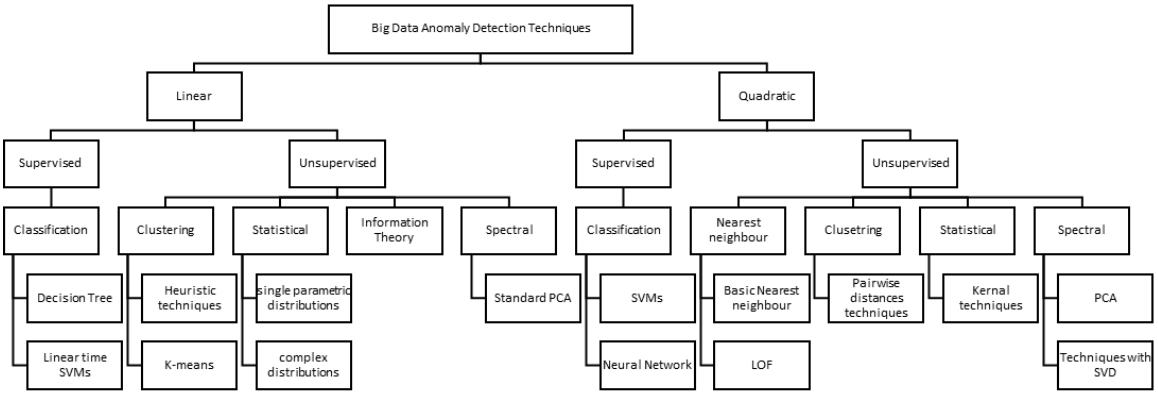

Figure 3.1: Velocity - Time Complexity Classification

Each category in time complexity uses both techniques; supervised and unsupervised. In the linear time, the linear SVM and decision tree under the classification techniques are examples of linear supervised techniques. The unsupervised techniques for linear time include clustering, statistical, information theory, and spectral. On the contrary, quadratic supervised techniques have SVM and neural network classifiers. Similar to linear unsupervised techniques, quadratic unsupervised techniques have four types; nearest neighbour, clustering, statistical, and spectral.

3.2 Variety - Data Nature Classification

There are several types of data that can affect the classification of anomaly detection techniques as shown in Figure 3.2. In general, the data has three types based on the data structure. 1) Structured data is organized information that can be easily stored, entered, and analyzed, 2) Semi-structured data is semi-organized information that has some sort of properties, and 3) Unstructured data is not organized information such as free documents or files. Under these three data types, the Big Data sources are listed with many examples.

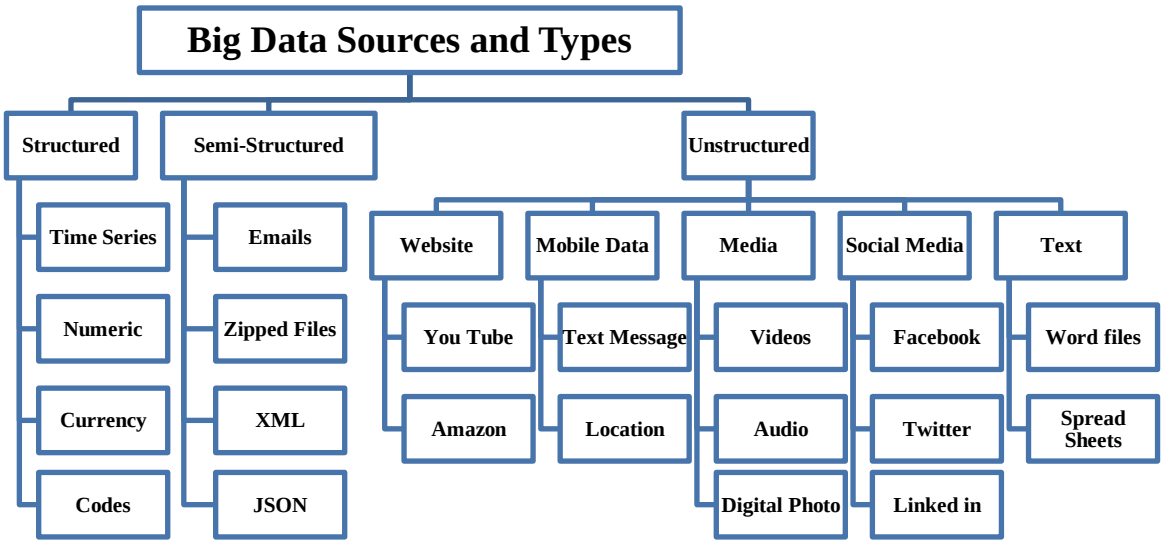

Figure 3.2: Big Data Sources and Types

Anomaly detection can be grouped into four categories based on the nature of the data because the variety of Big Data will affect the algorithm type, which is shown in the previous figure. These four categories are the most popular data sources which are time

series, text, social media, and media. Every data source has some commonly used anomaly detection technique as shown in Figure 3.3.

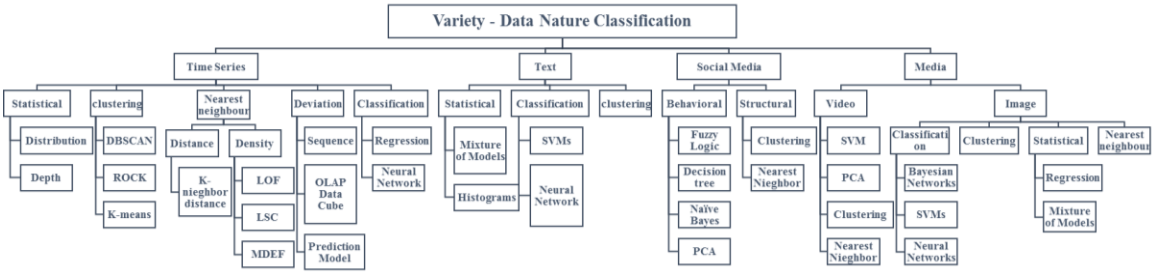

Figure 3.3: Variety – Data Nature Classification

Time series, under the structured data as explained before in figure 4, includes five popular anomaly detection techniques; statistical, clustering, nearest neighbour, classification, and deviation. For every type, there are several examples. Unstructured data has many important sources; however, the major source is chosen. Text source is one of the major unstructured data sources that has many relations for other sources such as mobile data and websites. The text data have statistical, classification, and clustering anomaly detection techniques. Also, social media is an unstructured data source that has several anomaly detection techniques based on behavioural and structural approaches. Likewise, media sources are an important unstructured data source which will be divided into image and video data. Image data varies with four anomaly detection techniques; classification, clustering, statistical, and nearest neighbour. Video data includes nearest neighbour, clustering, and some classification techniques such as SVM and neural network.

3.3 Volume - Data Feature Classification

The anomaly detection techniques can be broken into two major categories based on feature types, because the volume of Big Data will affect the anomaly detection techniques; univariant and multivariant techniques as shown in Figure 3.4. Under each feature type, there are two data types; discrete and continuous.

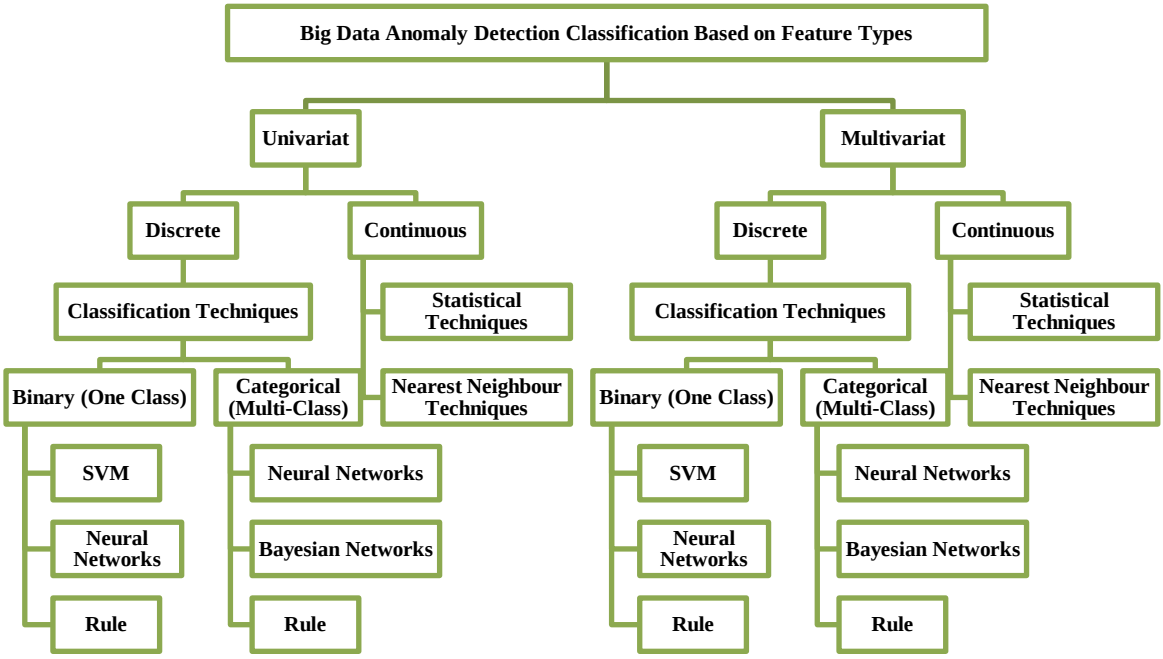

Figure 3.4: Volume - Data Feature Classification

The classification techniques under the discrete type will be divided into one class and multi-class for both feature types. On the other hand, the continuous data will have statistical and nearest neighbour techniques for both feature types.

3.4 Comparison Study

A comparative study of support vector machines and neural network techniques will be presented. We will compare between the techniques based on selected factors which will allow researchers to drive critical thinking ideas such as choosing a suitable model for certain problems and conditions. The criteria of choosing the research papers depend on two shared factors: the approach type (SVM or NN) and anomaly detection problem. The result of this study is expressed in Table 3.2 for SVM and neural network respectively. Only Neural Network will be implemented in this thesis because it will be suitable for our application. However, SVM has been researched in term of helping researchers choosing the best model regarding their problems. Where AUC represents **A**rea **U**nder **C**urve.

Table 3.2: SVM and Neural Network Comparison Table

Kernel Type	Problem Domain	Accuracy	NN Type	Problem Domain	Accuracy
Gaussian kernel [13]	Real-World System Call	0.953 (AUC)	Feed Forward [47]	Benchmark Network	0.958 (detection)
Linear kernel [39]	Wifi 802.11 Networks	0.982 (classification)	RNN [48]	Cyber-Physical Systems	N/A
Linear kernel [41]	Wireless Sensor Networks	0.971 (detection)	3D CNN [49]	Video Data	N/A
Gaussian kernels [42]	Petroleum Industry	N/A	FeedForward [50]	Driver Identification	0.81(overall)
Gaussian kernels [43]	Earth Dam and Levee	0.96 (F1-score)	MLP [51]	Electro-cardiogram	0.99 (classification)
Gaussian kernel [44]	Geological	0.8773 (AUC)	MLP [52]	Local ISP Network	0.96(detection)
Gaussian kernels [45]	Soft Computing	0.9995 (overall)	ANN [53]	Planting Calendar	0.846 (prediction)
Gaussian kernels [46]	Radar Imagery	0.97(overall)	RNN [54]	Web Applications	0.97 (detection rate)

A comparative study of K-means Clustering, HMM, Auto-Encoder Neural Network, and Gaussian Distribution will be presented. We will compare between the techniques based on selected factors which will allow researchers to drive critical thinking ideas such as choosing a suitable model for certain problems and conditions. The criteria of choosing the research papers depend on two shared factors: the approach type (K-means, HMM, NN, or GD) and anomaly detection problem. The result of this study expressed in Table 3.3 for K-means, HMM, Auto-Encoder, and Gaussian Distribution respectively. DA is the detection accuracy. All the models will be implemented for a comparison task.

Table 3.3: K-means, HMM, Auto-encoder, and Gaussian Distribution Comparison Table

K-means Clustering			**HMM**		
Problem Field	**Cluster Number**	**DA/%**	**Problem Field**	**States Number**	**DA/%**
Network attack[55]	50	96	Health care [63]	107	95
Network attack [56]	2	93.9	Home activity[64]	10	87
Network attack [57]	8	98	computer systems [65]	3	91.578
Network attack [58]	60	81	computer system [66]	2	90
Network attack[59]	100	80.119	Network [67]	2	92.25
Network attack [60]	5	92	computer network [68]	20	86
Cloud Computing [61]	26	96.44	System Calls [69]	6	81.7
Smart Grid [62]	3	91	Cognitive Radio [70]	4	80
Auto-Encoder			**Gaussian Distribution**		
Problem Field	**Encoder Type**	**DA/%**	**Problem Field**	**Gaussian Type**	**DA/%**
Web Attacks[71]	Stacked	88.34	School Electricity Consumption [79]	Combined-regression	89
System Logs[72]	Convolutional	94	Dictionary Learning [80]	background	94
computer vision [73]	Deep	97	Network [81]	Graphical	86
network monitoring [74]	Variational	95	Hyperspectral image processing[82]	Multi-dimensional	91
Credit Card Transactions [75]	Combined-OCSVM	96.85	Gas Turbine Engine [83]	Combined-Deep Learning	99.75
Video and localisation [76]	sparsity and reconstruction	82	Bankruptcy [84]	multivariate	89
infrared spectroscopy [77]	Stacked	95	Network attack [85]	Mixture	99.39
Negative Health Events [78]	LSTM	87	hyperspectral imaging [86]	SMV-SCM	93

3.5 Summary

The three classifications that are provided in this chapter cover anomaly detection techniques in Big Data applications. These classifications inspired us to build an anomaly detection system using combination models of the machine learning techniques that are in the classifications. The next chapter will explain in detail the proposed anomaly detection system.

Chapter 4

4 Proposed Anomaly Detection System

This chapter proposes an anomaly detection system in novel combination models containing machine learning techniques. The combination models rely on several unsupervised techniques for the same reasons that are mentioned in Chapter 2. Figure 4.1 lists all machine learning techniques that are used and their purposes. Parameter tuning step will be explained in every model. In addition, this chapter will explain the common evolution methods as well as the proposed sequential evaluation algorithm to evaluate the model in a very accurate way. This chapter will also provide a detailed discussion and comparison between all the models and present evaluation methods including the final and best results. Finally, a chapter summary will recap the most important outcomes in this chapter to utilize these outcomes in the next authentication step.

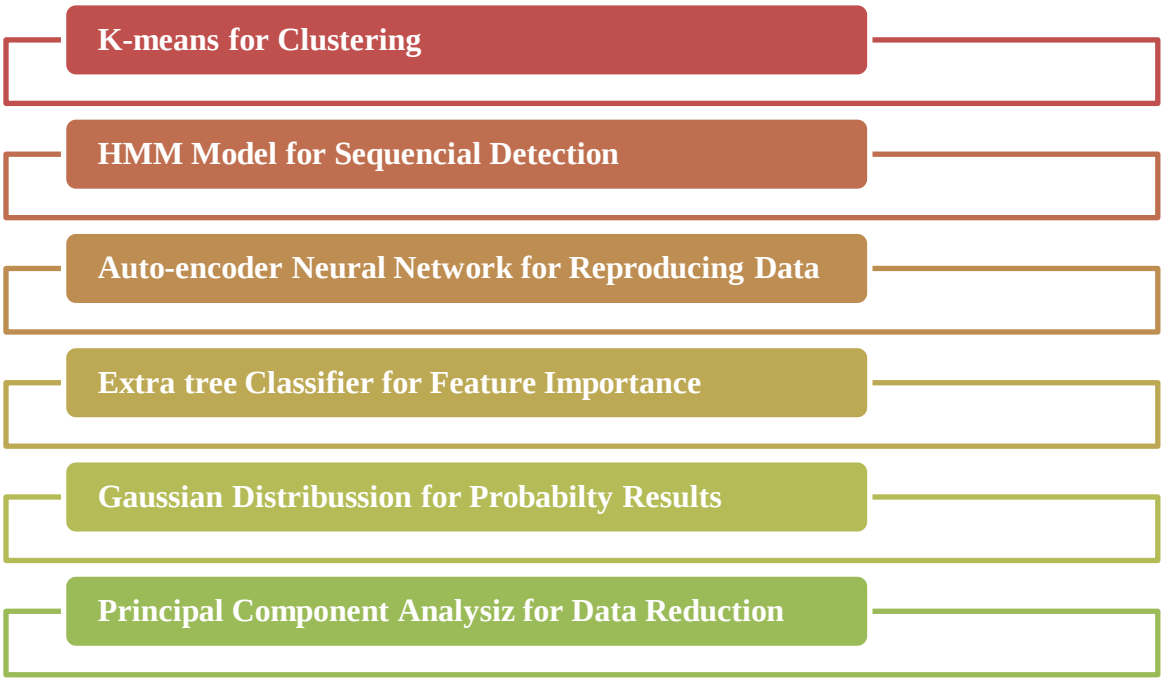

Figure 4.1: Used Machine Learning Techniques and their purposes

4.1 General Architecture

Anomaly detection systems, in general, have three known steps: first, to choose and prepare the most inductive features for anomalous observations. Secondly, fitting the technique parameters to learn the normal behaviour. Lastly, to feed the new examples to the technique for the detection process. Overall, the proposed anomaly detection system architecture can be divided into five parts in Figure 4.2.

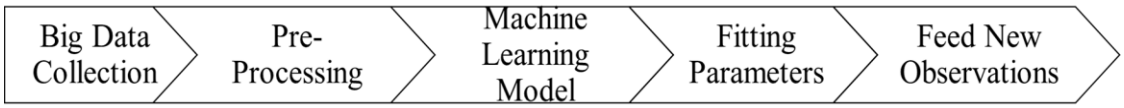

Figure 4.2: Anomaly Detection Proposed Architecture

In this research, we assumed that the data is collected from different Big Data sources. Prior to anomaly detection processing, there is a preprocessing step if the data needs to be preprocessed. Normalization is one of the data preparing steps that makes the data values in one scale to have more accurate results. There are several methods to normalize the data. However, mean normalization is an efficient method to normalize the attribute values through the following equation:

$$\boldsymbol{X} = \frac{\boldsymbol{x}-\boldsymbol{\mu}}{\boldsymbol{s}} \qquad (4.1)$$

Where x is the input data attributes, μ is the mean value, and s is the standard deviation value. The second preparing step is categorizing which attributes need to be categorized before the processing step because it contains text information, or if it is difficult to analyze. For example: if a gender attribute has two values in a dataset; Male 'M' and Female 'F'. We categorized it as 1 for male and 2 for female.

Due to the massive amount of data, anomalous patterns will not be clear with a lot of normal patterns. As a result, dimensional reduction is one of the vital preparing methods which can be done using many techniques. The Principle Component Analysis (PCA) technique is a prevalent method for this preparing step (dimensional reduction). This aggregates the data attributes into smaller attributes. Moreover, it assumes that the data is

a matrix m-by-n dimension. Each row in the matrix determines feature values for a user in a time stamp. Formally, PCA is a projection method that maps a given set of data points onto principal components [3]. The first step is to convert the datasets into a matrix and find the relationships among the features by calculating equation 4.2.

$$\boldsymbol{c = X' * X} \qquad (4.2)$$

Where X is the input data and C is called a covariance matrix. Then, we will find the eigen values and eigen vectors of the covariance matrix sigma and sort them decreasingly which is also called an eigen decomposition. The eigen values (W) is the variance in the dataset and eigen vectors (Lambda) is the corresponding direction of the variance. After that, we will select a number of W corresponding to Lambda which is 2 in our problem. The data features will be reduced based on this number. The last step is to calculate the reduced data by multiplying the Lambda with only two vectors with datasets. We have this shown in equation 4.3.

$$\boldsymbol{Reduced\ Data = X * W} \qquad (4.3)$$

Lastly, the data will be split into train, cross-validation, and test sets, as shown in Table 4.1. The training dataset will only have 60% normal observations and no abnormal observations to learn the technique different than the normal patterns. The cross-validation and test datasets will have 20% of the normal observations, and the abnormal observations will be split equally between them to feed the new abnormal observations and evaluate the detection.

Table 4.1: Data Splitting in Anomaly Detection System

Datasets	Normal Observations	Abnormal Observations
Train set	60%	0%
Cross validation set	20%	50%
Test set	20%	50%

Then, a training anomaly detection system using several machine learning models will be used as unsupervised techniques to assign the anomalous data. After that, an evaluation method will be used to calculate the model's accuracy.

4.2 Anomaly Detection - Machine Learning Models

An anomaly detection problem is a binary classification problem in terms of machine learning problems. So, the final results will be varied between 0s and 1s. To obtain the best result, a comparison will be provided in the discussion section between the three machine learning models. The understanding of model usage is crucial to enhance the discussion. The model's usage is explained in the next subsections.

4.2.1 K-means Clustering, HMM, and Auto-encoder Models

This model utilizes K-means clustering to assume that the big data has several clusters and assigns random centroids positions for every cluster based on observations concentration. This model can work with one cluster or more. In the case of one cluster, the technique will assign one centroid for the whole data then several steps can be taken. For example, the threshold distance value from the centroid will be flagged as an anomaly. In the case of two clusters, it can be done in numerous ways; it could be one cluster for the normal data and the other cluster for the abnormal data or it could be two clusters for normal and threshold distance values from the centroids will be flagged as an anomaly. In three or more cluster cases, the data will have more than two clusters which means a threshold value should be considered or one cluster will be for anomalies and the others will be normal instances. Figure 4.3 shows the general workflow of this model where Big Data is fed to a K-means clustering technique. Then, the final binary production will be generated directly from K-means or through threshold values.

In HMM model, we will use the Hidden Markov Model for predicting the anomalous data in sequential form as shown in Figure 4.3. Two states will be utilized to assign one for normal observations and the other for abnormal observations. HMM needs a probability matrix that will be assumed based on the data distribution. The output or final predictions will contain 0's for normal and 1's for abnormal observations.

The Auto-encoder model has at least three layers (input, hidden, and output) of neural networks to reproduce the input data and learn the normal behaviour. The number of neurons in every layer will be tuned related to the input observation number. Then a threshold value will be user based on the reconstruction error. If the data exceeds this threshold value it will be flagged as an anomaly otherwise it will be normal as shown in Figure 4.3.

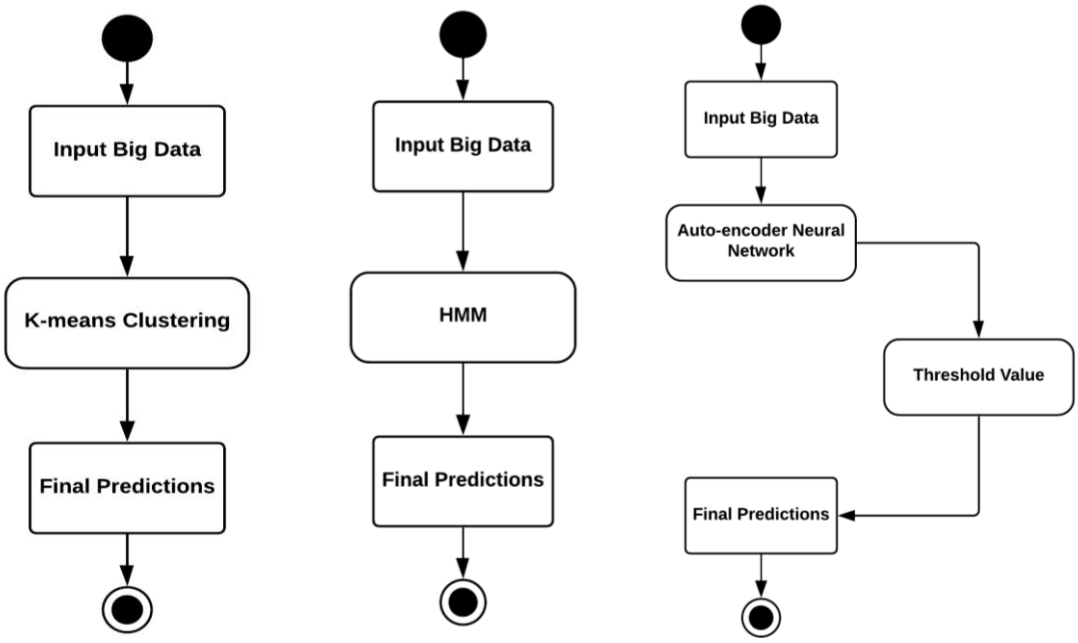

Figure 4.3: K-means Clustering, HMM, and Auto-encoder Models

4.2.2 Auto-Encoder-K-means and Auto-Encoder-HMM Models

In the Auto-Encoder with K-means model, a series combination between K-means and auto-encoder will be used as shown in Figure 4.4. The auto-encoder will be trained on the normal observations. Then K-means will work with the threshold of the previous section. Moreover, K-means will cluster the reconstructed data which is the output from the auto-encoder. The clusters will be two or one for the anomaly and normal for the rest of the clusters.

Auto-Encoder with HMM model uses the same combination of the previous one by replacing K-means with HMM. As mentioned in the prior section, HMM will use the reconstructed data that was produced by the auto-encoder to predict the anomalous

observations. This method will increase the HMM accuracy as it will be discussed later. HMM will use two states one for anomaly and one for normal data. Figure 4.4 shows the entire Auto-encoder and HMM model.

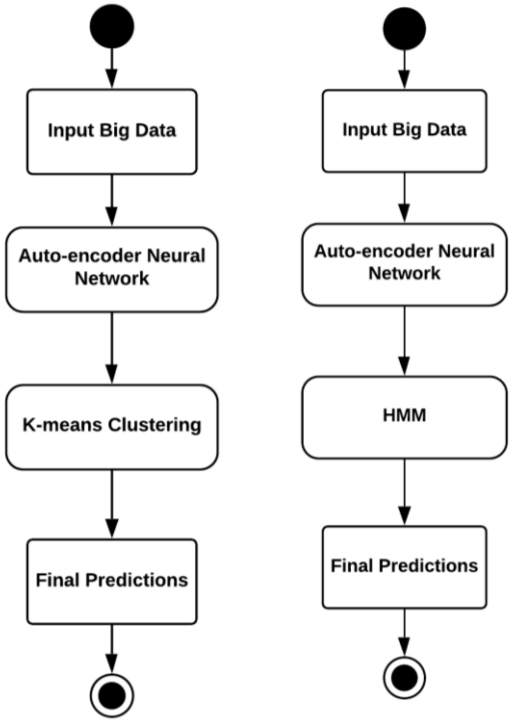

Figure 4.4: Auto-Encoder-K-means and Auto-Encoder-HMM Models

4.2.3 Combination Model (Auto-encoder, K-means, and HMM)

In this model, we will utilize all the previous techniques; auto-encoder, HMM, and K-means, in one combination. Figure 4.5 shows the diagram of this model. Auto-encoder will reproduce the data and send it to the HMM. HMM will predict the anomalous data using the reproduced data from the auto-encoder. The purpose of K-means clustering in this model is to calculate the probability matrices that HMM needs based on the data distribution. So, HMM will receive two inputs; one from auto-encoder, which is the reproduced data, and the other from K-means clustering, which is the probability values. K-means will also use the reproduced data from the auto-encoder.

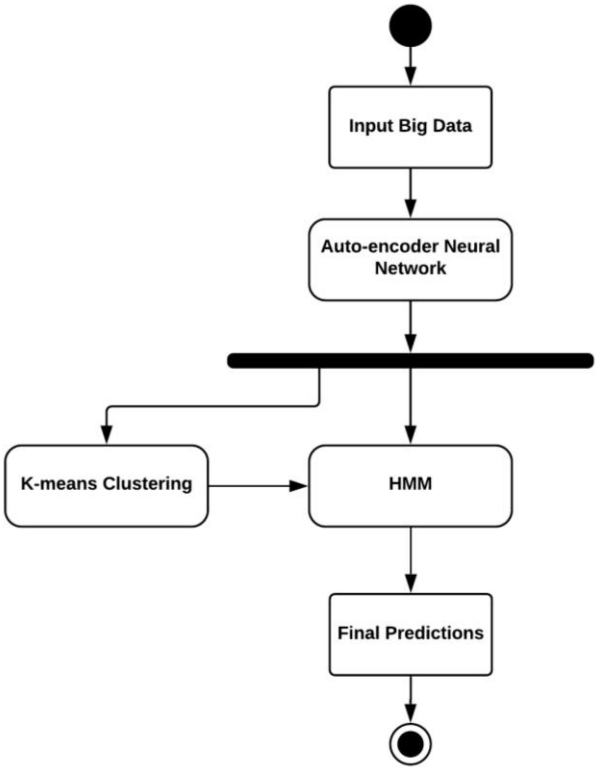

Figure 4.5: Auto-Encoder, K-means, and HMM Model

4.2.4 Gaussian Distribution Model

This model totally relies on the populistic Gaussian distribution. The model will be trained and learns the probability values of the normal observations. Then the model will be fed with new data which has anomalous data to detect them through a threshold probability value. Cross validation will be used after the model is built. The test set will be used as a final feeding step.

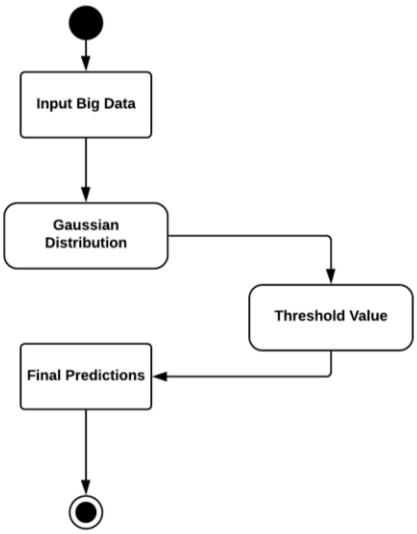

Figure 4.6: Gaussian Distribution Model

4.3 Programming, Libraries and Evaluation Methods

In this section, all the models will be trained and built using the training dataset. These models will be tuned using the cross validation set. Finally, the model will be tested using the test set. The splitting percentage is mentioned in Table 4.1.

4.3.1 Program Libraries

We used Python language to create our model and experiment with our datasets. The libraries and metrics described in Table 4.2 will be used relating to a specific model.

Table 4.2: Used Python Libraries and Description

Python Library Name	Usage	Description
• Pandas	preprocessing	pandas offer data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license.
• numpy	preprocessing	NumPy is adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level

		mathematical functions to operate on these arrays.
• Sklearn-learn: o Metrics: ▪ roc_auc_score ▪ balanced_accuracy_score ▪ mean_squared_error ▪ accuracy_score, ▪ f1_score, ▪ precision_score, ▪ recall_score, ▪ classification_report, ▪ confusion_matrix , o model_selection: ▪ train_test_split, o ‘preprocessing: ▪ Scale StandardScaler, ▪ decomposition: PCA, ▪ ‘math: sqrt , o Datasets: ▪ load_digits, ▪ ‘matplotlib.pyplot’, ▪ and ‘time’.	Evaluation	Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms. It also includes matrices and preprocessing operations for dataset
• sklearn.ensemble, ExtraTreesClassifier	Feature Importance	
• ‘sklearn.cluster’ KMeans	K-means: model	
• ‘hmmlearn’ hmm	Hmm model	Simple algorithms and models to learn Hidden Markov Models in Python. It follows scikit-learn API as close as possible, but adapted to sequence data. It built on scikit-learn, NumPy, SciPy, and matplotlib. It is Open source, commercially usable with BSD license.
• Tensorflow o scipy o stats, o seaborn, o pickle, o pylab o rcParams, • keras.models o Model,	Auto-encoder model	TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks.

○ load_model ○ keras.layers ○ Input, • Dense ○ keras.callbacks ○ ModelCheckpoint, ○ TensorBoard ○ Keras import regularizers,		

4.3.2 Common Evaluation Methods

Binary classification has many evaluation methods. One of the popular methods is a confusion matrix to calculate the classification accuracy. The accuracy equation of the confusion matrix as equation 4.4 explains requires a calculation for many variables. These variables are True-positive, True-negative, False-positive, and False-negative. True-positive is the number of observations that are actually normal instances, and the technique predicts it as normal instances (i.e. the number of items correctly labeled as belonging to the positive class). True-negative is the number of observations that are the actual abnormal instances and the technique predicts it as abnormal instances. False-positive is the number of observations that are the actual is abnormal instances, but the technique predicts it as normal instances (i.e. the sum of true positives and false positives, which are items incorrectly labeled as belonging to the class). False-negative is the number of observations that are the actually normal instances, but the technique predicts it as abnormal instances. All of these variables are summarized in Table 4.3.

$$\boldsymbol{Accuraccy} = \frac{\boldsymbol{TP+TN}}{\boldsymbol{TP+TN+FP+FN}} \qquad (4.4)$$

Table 4.3: Confusion Matrix Table

		Actual Values	
		Positive	**Negative**
Predicted Values	**Positive**	True-Positive (TP)	False-Positive (FP)
	Negative	False-Negative (FN)	True-Negative (TN)

From the confusion matrix, more variables can be calculated to give more accurate insights, especially with unbalanced data such as in our case. Precision or positive

predictive value, in binary classification, is the fraction of true positive observations among the total number of positive observations; true and false as shown in equation 4.5.

$$\boldsymbol{Precision} = \frac{\boldsymbol{TP}}{\boldsymbol{TP+FP}} \qquad (4.5)$$

Equation 4.6 shows that the recall, true-positive rate, or sensitivity is the number of true positive observations divided by the total number of true positive and false negative observations combined. Both precision and recall give more understanding and measure of relevance.

$$\boldsymbol{Recall} = \frac{\boldsymbol{TP}}{\boldsymbol{TP+FN}} \qquad (4.6)$$

F1-score is an accuracy measure that considers both precision and recall getting the accurate results in term of unbalanced data in machine leaning models. That means F1 score is the harmonic average that varies from 1 to 0. Therefore, an F1 score of 1 is considered a perfect model, while an F1 score of 0 is a total failure. In more detail, if a model has a good F1 score that means it has low false positive and negative observations. So, the model is correctly identifying real anomalies and there are no false alarms for this model. Equation 4.7 explains how an F1 score is the multiplication of precision by recall divided by the summation of them and the result will be multiplied by 2.

$$\boldsymbol{F_1} = \boldsymbol{2} \cdot \frac{\boldsymbol{Precision \cdot recall}}{\boldsymbol{precision+recall}} \qquad (4.7)$$

Receiver operating characteristic curve (ROC) demonstrates the binary classification model's accuracy and ability in graphical plots. The ROC curve is plotted using the true positive rate (TPR) against the false positive rate (FPR) at various threshold values.

Also, we will use the misclassification error which calculates the error in a percentage format. Equation 4.8 shows the relation for this error.

$$\boldsymbol{Missclassification\ Error} = \boldsymbol{1} - \left(\frac{\boldsymbol{sum(original\ classes)}}{\boldsymbol{sum(predicted\ classes)}}\right) \qquad (4.8)$$

Where the original classes are the classes that are given in the test dataset and the predicted classes are the classes that are presented by the technique. Then, the error will be converted to percentage format. Additionally, root mean square error will be determined for each technique with equation 4.9.

$$\mathbf{RMSE} = \sqrt{\mathbf{mean}((\mathbf{Orignal\ values} - \mathbf{Predicted\ values})^{\mathbf{2}})} \qquad (4.9)$$

Where the original values are the values that are given in the test dataset and the predicted values are the values that are given by the technique. True-Positive Rate (TPR) and True-Negative Rate (TNR) are calculated from the confusion matrix using equations 4.10 and 4.11.

$$\boldsymbol{TPR} = \mathbf{tp}/(\mathbf{tp} + \mathbf{fn}) \qquad (4.10)$$

$$\boldsymbol{TNR} = \mathbf{tn}/(\mathbf{tn} + \mathbf{fp}) \qquad (4.11)$$

We need to develop three sequential accuracy algorithms for true positive rate, true negative rate and the accuracy to make sure that the pervious evaluation methods are not only calculating the predicted observation numbers but also matching the instances between the original and the predicted values.

4.3.3 Sequential Accuracy Algorithm (SAA)

The following algorithms are written in seeking efficiency and certainty. The first one will compute the overall accuracy based on a sequential tracking for every user between anomaly and abnormal cases.

Algorithm 1: All data sequential accuracy
INPUT: binary prediction for "Class" feature
OUTPUT: percentage accuracy for the whole data predictions

```
Begin
    Read the input data from the model output(predictions)
    Read the original labels from data(y_actual)
    Create "C" Data frame
    If predictions equal to y_acual then
        Add 1 to "C"
    Else
        Add 0 to "C"
    End if
    Count 1 number in "C"
    Divide the number of one's by the data length
    Multiply the result by 100
    Show the output accuracy
End
```

The second algorithm will compute the accuracy for only the normal instances based on sequential tracking for every user in the related target normal cases.

Algorithm 2: Normal data sequential accuracy
INPUT: binary predictions for only normal observations in the "Class" feature
OUTPUT: percentage accuracy for the normal data predictions

```
Begin
    Read the input data from the model output(predictions)
    Read the original labels from data(y_actual)
    Extract only the zeros on y_actual and the related predictions to "s"
    Create "C" Data frame
    If predictions in "s" equal to y_acual "s" then:
        Add 1 to "C"
    Else
        Add 0 to "C"
    End if
    Count 1 number in "C"
    Divide the number of one's by the data length
    Multiply the result by 100
    Show the output accuracy
End
```

The third algorithm will compute the accuracy for only the abnormal instances based on sequential tracking for every user in the related target abnormal cases.

Algorithm 3: Abnormal data sequential accuracy
INPUT: binary predictions for only abnormal observations in the "Class" feature
OUTPUT: percentage accuracy for the abnormal data predictions

```
Begin
    Read the input data from the model output(predictions)
    Read the original labels from data(y_actual)
    Extract only the ones on y_actual and the related predictions to "s"
    Create "C" Data frame
    If predictions in "s" equal to y_acual in "s" then:
        Add 1 to "C"
    Else
        Add 0 to "C"
    End if
    Count 1 number in "C"
    Divide the number of one's by the data length
    Multiply the result by 100
    Show the output accuracy
End
```

All three previous algorithms were applied to ensure that the known evaluation metrics are calculating the exact user accuracy based on the target feature. The first algorithm matched the same results of the accuracy based on the confusion matrix library in Python. The second algorithm matched the same result that the true positive rate generated out of the accuracy metrics in Python. The third algorithm gave the same result compared to true negative rate out of the accuracy metrics.

4.3.4 Parameters Tuning

This section explains the parameters that we tried to tune in all the techniques. Some parameters have fixed values. But other parameters have a wide range to tune. In this case the parameter will be tuned on the wide range in general over a fixed value and then will be focused on the higher small ranges. In Table 4.4 the tuning parameters are described through the input type and Python indication name for every model separately. The definition of every parameter is provided from the Python website.

Table 4.4: Tuning Parameters in Python

K-means Tuning Parameters	**Python Indication Parameter**	**Input Type**
Number of Clusters	"n_clusters"	Integer
Random State	"RandomState"	Integer
Algorithm	"algorithm"	String
Tolerance	"tol"	Float
Initialization Method	"init_"	String
Maximum of Iteration	"max_iter"	Integer
Initialization Number	"n_init"	Integer
Number of Jobs	"n_jobs"	Integer
HMM Tuning Parameters	**Python Indication Parameter**	**Input Type**
Number of Components	"n_components"	Integer
Covariance Type	"covariance_type"	String
Covariance Minimum	"min_covar"	Float
Algorithm	"algorithm"	String
Random State	"random_state"	Integer
Number of Iterations	"n_iter"	Integer
Tolerance	"tol"	Float
Auto-Encoder Tuning parameters	**Python Indication Parameter**	**Input Type**
Activation function	"activation"	String
Hidden layers and neurons number	Programmer assign "hidden_dim"," encoding_dim"	Integer
Number epoch	Programmer assign "nb_epoch"	Integer
Batch size	Programmer assign "batch_size"	Integer
Learning rate	Programmer assign "learning_rate"	Float
Threshold	Programmer assign "threshold"	Integer

K-means is a clustering technique that has several parameters under its library in Python. Number of clusters (n_clusters) is the number of clusters to form as well as the number of centroids to generate. Random State (random_state) determines random number generation for centroid initialization. 'None' is the default Python value for random state. Algorithm (algorithm) is the K-means algorithm to use such as "auto", "full" or "elkan". The classical expectation–maximization (EM)-style algorithm is "full". The "elkan" variation is more efficient by using the triangle inequality, but currently does not support

sparse data. “auto” chooses “elkan” for dense data and “full” for sparse data. Python K-means default algorithm is “auto”. Tolerance (tol) is relative tolerance with regards to inertia to declare convergence. Python tolerance default is 1e-4. Initialization method (init_) is the initialization methods such as {‘K-means++’, ‘random’ or ‘ndarray’}, the Python default is ‘K-means++’.

- ‘K-means++’ : selects initial cluster centers for k-mean clustering in a smart way to speed up convergence. See section Notes in k_init for more details.

- ‘random’: choose k observations (rows) at random from data for the initial centroids.

- If ‘ndarray’ is passed, it should be of shape (n_clusters, n_features) and gives the initial centers.

Maximum of Iteration (max_iter) is the maximum number of iterations of the K-means algorithm for a single run. The Python default is 300 iterations. Initialization Number (n_init) is the number of times the K-means algorithm will be run with different centroid seeds. The final results will be the best output number of initialization consecutive runs in terms of inertia. The Python default is 10 times. Number of jobs (n_jobs) is the number of jobs to use for the computation. This works by computing each of the initialization number runs in parallel. None means 1 unless in a joblib.parallel_backend context. -1 means using all processors. See Glossary for more details. ‘None’ is the default Python value for number of jobs.

HMM model has several types and under every type there are several parameters. GaussianHMM is the chosen model in our simulation. GaussianHMM is a Hidden Markov Model with Gaussian emissions. The number of components iterations (n_components) is a number of states. Covariance type (covariance_type) is a string describing the type of covariance parameters to use. It must be one of the following:

“spherical” — each state uses a single variance value that applies to all features.
“diag” — each state uses a diagonal covariance matrix.
“full” — each state uses a full (i.e. unrestricted) covariance matrix.

“tied” — all states use the same full covariance matrix. Defaults to “diag”.

Covariance minimum (min_covar) is a floor on the diagonal of the covariance matrix to prevent overfitting. The Python defaults in this parameter is 1e-3. Algorithm (algorithm) is a decoder algorithm. It must be one of the following algorithms “viterbi” or map”. Python algorithms defaults in this parameter is “viterbi”. Random State (random_state) is a random number generator instance. Number of iterations (n_iter) is a maximum number of iterations to perform. Tolerance (tol) is a convergence threshold. EM will stop if the gain in log-likelihood is below this tolerance value. “hmm.GMMHMM” is a Hidden Markov Model with Gaussian mixture emissions. “hmm.MultinomialHMM” is a Hidden Markov Model with multinomial (discrete) emissions.

The parameters of Auto-encoder Neural Network are many. However, some of these parameters have been tuned and explained based on its effects. Activation function (activation) is an activation function to use. The activation functions are: “Softmax” is Softmax activation function. “elu” is Exponential linear unit. “selu” is Scaled Exponential Linear Unit (SELU). “softplus” is Softplus activation function. “softsign” is Softsign activation function x / (abs(x) + 1). “relu” is Rectified Linear Unit. max(x, 0). “tanh” is Hyperbolic tangent activation function. “sigmoid” is Sigmoid activation function. “hard_sigmoid” is Hard sigmoid activation function. “exponential” is Exponential (base e) activation function. “linear” is applied (a(x) = x). Python default activation function is linear. Hidden layers are the number of neurons in every specified hidden layer such as (hidden_dim1 = 5). Number epoch is the number of iterations that the auto-encoder will run. Batch size is the number of examples from the training dataset used in the estimate of the error gradient. Learning rate is a float number that is related to the algorithm convergence step. Threshold is a value that will divide the dataset into different groups usually based on error.

4.4 Anomaly Detection Results

These results are divided based on the three models that were described in the previous section. The best results are presented in tables that are chosen out of many tuning

results regarding some parameters. The evaluation methods that we focus on are only the true positive rate and true negative rate because of three reasons:

a) Imbalanced data between normal and abnormal observation numbers.

b) Our proposed user authentication system requires high accuracy in the abnormal detection accuracy to use the results correctly in the next step.

c) These methods will give us an indication for the abnormal and normal detection accuracies separately.

4.4.1 Experiment 1 - Credit Card Dataset

Experiment one was implemented on a credit card dataset. The dataset contains transactions made by credit cards in September 2013 by European cardholders. Some Big Data characteristics are applied to this dataset such as samples volume with respect to the time and features variety. The original dataset presents transactions that occurred in two days in 284807 observations with 31 variables. The dataset is divided into three sets; train, cross-validation, and test sets.

Furthermore, the features of this data are time, amount of money, class, and set of unknown features. V1 to V28 features are the principal components obtained with PCA, but unfortunately, due to confidentiality issues, the original features' names and more background information about these features are unknown. All the features used as a numerical input (independent) variables are the time, amount, and V1 until V28. Some of the input features that are not normalized have been normalized. Class feature is only used for the evaluation part because it has data labels. Table 3.1 describes some of the dataset characteristics.

Table 4.5: Dataset 1 Description

Dataset name	Credit card dataset
Dataset features number	30
Dataset observation number	287456
Dataset Date	2013
Dataset place	Europe
Normal - Anomalous percentage	99.83 - 0.17%

To visualize the dataset, the histogram function in Python was applied on the dataset, four features is shown in Figure 4.7 as sample:

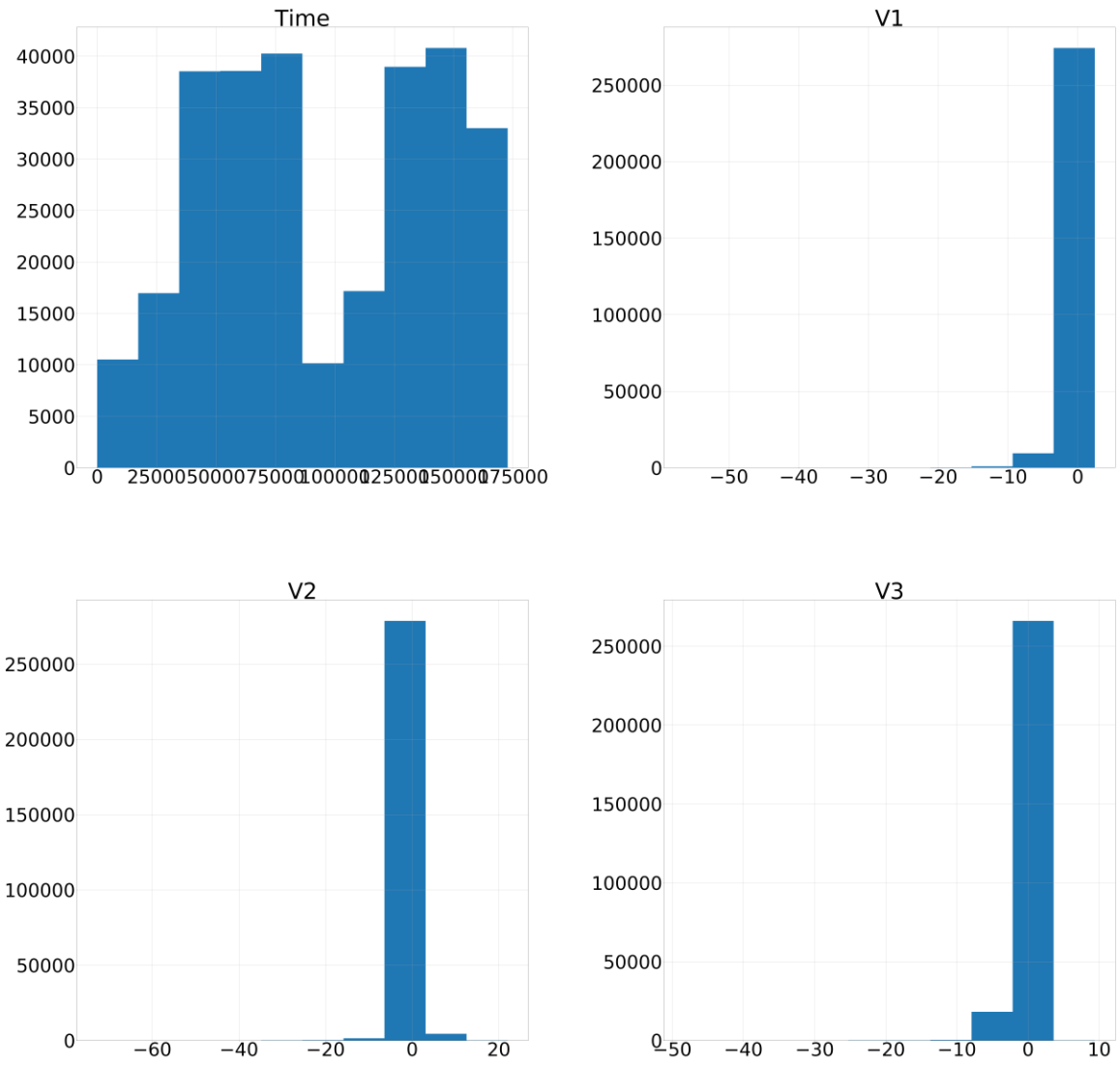

Figure 4.7: Features Histogram for Dataset 1

The dataset was already prepared and ready to use i.e. there are no NAN values, all features are numbers. Only some feature engineering is used to replace some features. For example, taking a log of one feature or multiplying it by a number to have a data close to a Gaussian distribution. Finally, feature importance was applied for applying PCA dimensional reduction. The features ware sorted in term of importance to the target using extra tree classifiers as shown in Figure 4.8. Additionally, a comparison is provided between data features in Appendix E.

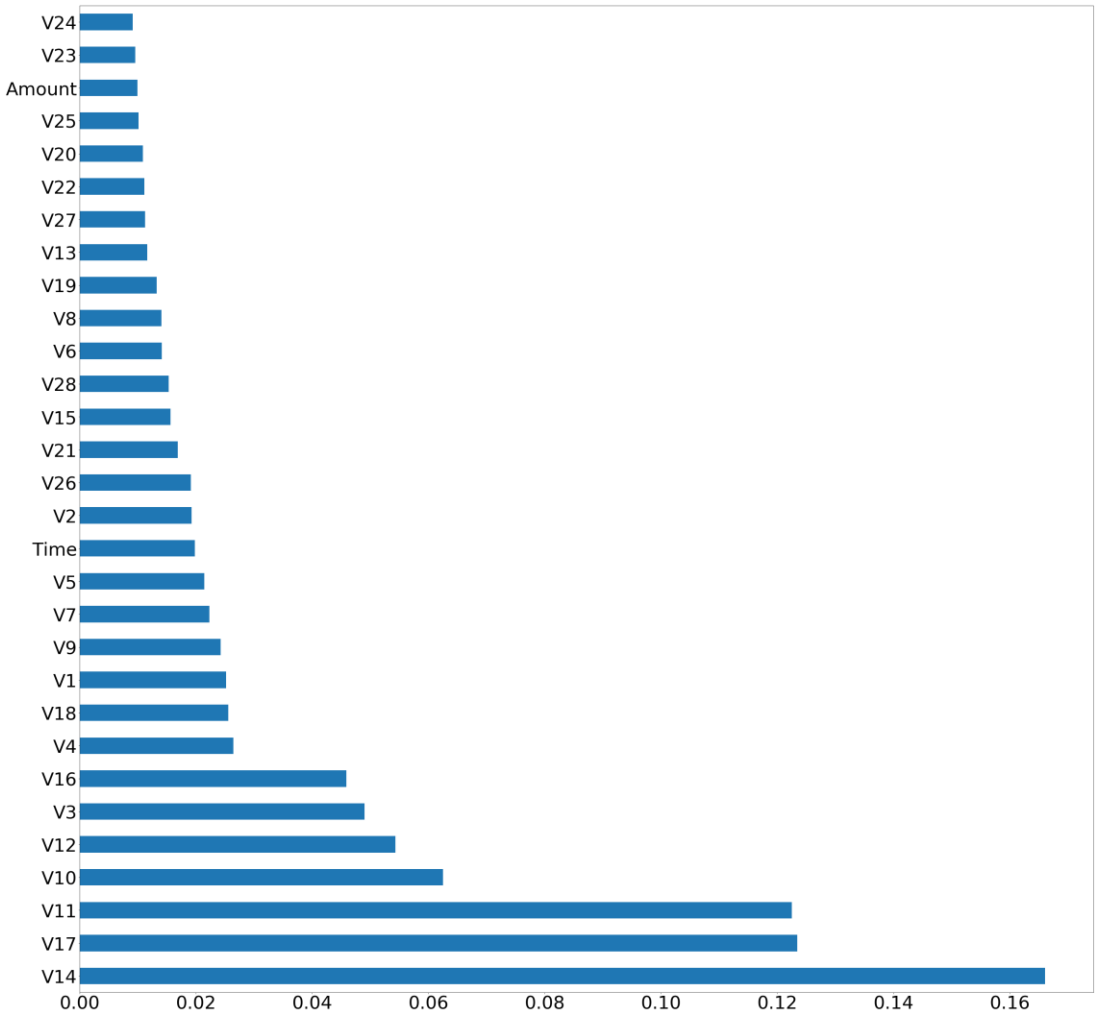

Figure 4.8: Feature Importance for Dataset 1

Default proposed models ware applied between four assumptions for comparison between normalization and dimensional reduction. Table 4.6 shows the results with the four assumptions for Dataset 1. In the K-means model, the best result for TPR of 71% was by applying the normalization and dimensional reduction assumption. However, the best TNR of 53% was in assumption one. The TNR in the fourth assumption which gave the highest TPR was not that far from the best one. So, the fourth assumption was chosen to be applied for tuning parameters.

In the HMM model, Table 4.6 shows that the best result for TPR of 91% was with the first two assumptions with the best TNR of 84%. So, the first assumption was applied for tuning parameters.

In the Auto-Encoder model, the best result for TPR was with assumptions one and three by 100%. However, the best TNR of 98% was in assumptions two and four. The TNR in assumptions one and three gave the highest TNR has 0% and a very low TPR which is not acceptable. So, the second assumption was applied for tuning parameters because it has the highest TNR and acceptable TPR.

In the Gaussian Distribution model, the best result for TPR of 81% was with assumptions two and four. However, the best TNR of 99.7% was in assumption three. The TPR in the third assumption which gave the highest TNR was not very low which is not acceptable. So, the fourth assumption was applied for tuning parameters. Finally, for more results such as F1 score and RMSE, refer to Appendix A, Appendix B, Appendix C, Appendix D.

Table 4.6: Results for Dataset 1 based on Four Assumptions

Models	Accuracy	TPR	TNR
Assumption 1: without normalization or dimensional reduction			
K-means	0.5329	0.3293	0.5338
HMM	0.8431	0.9106	0.8428
Auto-encoder	0.0043	1	0
Gaussian	0.9889	0.2764	0.992
Assumption 2: with normalization only			
K-means	0.5256	0.2805	0.5266
HMM	0.8432	0.9106	0.8429
Auto-encoder	0.9816	0.6504	0.9831
Gaussian	0.9921	0.813	0.9929
Assumption 3: with dimensional reduction only			
K-means	0.5329	0.3293	0.533792
HMM	0.7751	0.8374	0.774792
Auto-encoder	0.0043	1	0
Gaussian	0.9946	0.2073	0.997995
Assumption 4: with Both normalization and dimensional reduction			
K-means	0.4745	0.7195	0.4734
HMM	0.1568	0.0894	0.1571
Auto-encoder	0.9831	0.0285	0.9873
Gaussian	0.9923	0.813	0.993

Some results have an outstanding accuracy in the normal instances and unfortunate abnormal detection accuracy such as 14 and 91 in random states. Another group of results

have the opposite; unfortunate normal accuracy and excellent abnormal detection accuracy, for instance, 90 random states. Some results have an acceptable normal accuracy and outstanding abnormal accuracy like 42 random state. Table 4.7 summarizes all K-means results. Finally, for more results such as F1 score and RMSE, refer to Appendix M.

Table 4.7: K-means Results for Dataset 1

Tuning Parameters		**Evaluations**		
Max Iter	**Random State**	**Accuracy**	**TPR**	**TNR**
1	0	0.5026	0.2642	0.5037
10	0	0.5256	0.2805	0.5266
1	42	0.4637	0.882114	0.461917
10	42	0.4744	0.7195	0.4734
1	1	0.5512	0.4756	0.5515
10	1	0.4744	0.7195	0.4734
1	2	0.2495	0.5285	0.2483
10	2	0.5256	0.2805	0.5266
1	3	0.5361	0.4919	0.5363
10	3	0.5256	0.2805	0.5266
1	4	0.5552	0.4837	0.5555
10	4	0.6102	0.4065	0.6111
1	5	0.6623	0.674797	0.662241
10	5	0.5256	0.2805	0.5266
1	13	0.779	0.695122	0.779364
10	13	0.4744	0.7195	0.4734
1	14	0.9905	0	0.994777
10	14	0.5254	0.2886	0.5264
1	90	0.0118	0.910569	0.007879
10	90	0.5256	0.2805	0.5266
1	91	0.9829	0.073171	0.986846
10	91	0.4744	0.7195	0.4734
1	200	0.3517	0.939024	0.349155
10	200	0.4744	0.7195	0.4734
1	250	0.951	0.260163	0.95396
10	250	0.47443	0.71951	0.47337
Best Result		0.990492	0.939024	0.994777

The tuned parameters are initialization methods, initialization number, maximum number iteration, K-means algorithm, and random state. Every parameter has a range of variations, as shown in Table 4.8.

Table 4.8: Parameters Ranges

initialization method	K-means++	Random	ndarray
maximum number iteration	1 – 100		
K-means algorithm	Auto	Full	elkan
random state	0- 500		

The results in this model show better detections than K-means results in terms of accuracy. It has higher accuracy for both normal and abnormal detection. The highest result for both normal and abnormal detection has a "spherical" covariance type. The other results ware varied with "diag" and "full" covariance type and gave a satisfactory accuracy level for both, as shown in Table 4.9. Finally, for more results such as F1 score and RMSE in this part, refer to Appendix N.

Table 4.9: HMM Results for Dataset 1

Tuning Parameters				**Evaluations**		
Covariance type	**N iter**	**algorithm**	**Tol**	**Accuracy**	**TPR**	**TNR**
Spherical	5k	viterbi	0.1	0.93	0.89	0.93
Diag	5k	viterbi	0.1	0.84	0.91	0.84
Tied	5k	viterbi	0.1	0.52	0.23	0.52
Full		viterbi		0.68	0.89	0.68
Spherical		viterbi		0.70	0.90	0.90
Diag		viterbi		0.16	0.09	0.16
Tied		viterbi		0.48	0.77	0.48
Spherical	5k	map	0.1	0.90	0.90	0.90
Diag	5k	map	0.1	0.16	0.09	0.16
Tied	5k	map	0.1	0.52	0.23	0.52
Full		map		0.32	0.11	0.32
Spherical		map		0.10	0.10	0.10
Diag		map		0.48	0.77	0.48
Tied		map		0.52	0.23	0.52
Spherical	5k	viterbi		0.07	0.11	0.07
Spherical	5	viterbi	0.1	0.22	0.07	0.22

The auto-encoder results were tuned using the following parameters: number of epochs, batch size, input dimension, encoding dimension, hidden dimension for layer 1, hidden dimension for layer 2, activation function, learning rate, and threshold. The best results were obtained with varying the threshold values, as shown in Table 4.10. The highest abnormal detection accuracy has one threshold value, but the normal detection

accuracy has the lowest value. The threshold value of 2 has outstanding accuracy in both abnormal and normal accuracies. The other values have excellent accuracy for normal detection but an acceptable accuracy for abnormal detection. Overall, auto-encoder was better than both previous models, as shown in Table 4.10. This model has the best result, which was 0.88 abnormal detection accuracy and 0.95 normal detection accuracy of 2 threshold. Finally, for more results such as F1 score and RMSE in this part, refer to Appendix O.

Table 4.10: Auto-Encoder Model Results

Tuning Parameters					**Evaluations**	
Encoding _dim	**Hidden _dim1**	**Hidden _dim2**	**Activation**	**Threshold**	**TPR**	**TNR**
18	10	6	tanh	4	0.752	0.982
18	10	6	tanh	4	0.699	0.983
32	16	8	tanh	4	0.695	0.984
10	5	2	tanh	4	0.781	0.981
5	2	1	tanh	4	0.805	0.979
5	3	1	tanh	4	0.752	0.979
50	20	10	tanh	4	0.691	0.985
5	2	1	sigmoid	4	0.768	0.977
5	2	1	hard_ sigmoid	4	0.760	0.977
5	2	1	exponential	4	0.760	0.977
5	2	1	linear	4	0.756	0.981
5	2	1	tanh	3	0.825	0.972
5	2	1	tanh	2	0.878	0.954
5	2	1	tanh	1	0.923	0.836
5	2	1	tanh	5	0.655	0.984
5	2	1	linear	4	0.756	0.981
5	2	1	tanh	4	0.650	0.983
5	2	1	tanh	4	0.659	0.983
5	2	1	tanh	4	0.667	0.983

The results of the rest of the models are shown in Table 4.11. Auto-Encoder with K-means model did not give more accuracy from the auto-encoder model. However, there was good enhancement comparing with the K-means results, especially TPR, which is important in this research. Auto-Encoder with HMM model does not gave better results because the HMM results are already working well in term of TNR and TPR. The combination model between the three model (K-means, HMM, and Auto-Encoder) gave

better results compared to K-means and Auto-Encoder results. Comparing these results with HMM results indicates that there was little improvement between both TNR and TPR. Finally, Gaussian Distribution model reached the highest TNR with an acceptable TPR which shows that the Gaussian distribution model has a high ability to classify normal instances.

Table 4.11: Results of Four Models

Evaluations						
K-means with Auto-encoder Model Results						
Accuracy	**Precision**	**Recall**	**F1-score**	**RMSE**	**TPR**	**TNR**
0.5284	0.4971	0.3321	0.3468	0.6868	0.1341	0.5301
0.4716	0.503	0.6739	0.3266	0.7269	0.878	0.4698
HMM with Auto-encoder Model Results						
Accuracy	**Precision**	**Recall**	**F1-score**	**RMSE**	**TPR**	**TNR**
0.4914	0.4995	0.4694	0.3328	0.7132	0.4472	0.4916
K-means, HMM, and Auto-encoder Model Results						
Accuracy	**Precision**	**Recall**	**F1-score**	**RMSE**	**TPR**	**TNR**
0.5084	0.5023	0.6317	0.3429	0.7011	0.7561	0.5074
0.9794	0.4994	0.4979	0.4973	0.1434	0.0122	0.9836
0.4697	0.5031	0.6791	0.3258	0.7282	0.8902	0.4679
0.9304	0.5262	0.9125	0.5318	0.2637	0.894309	0.930605
0.8981	0.5183	0.9023	0.5087	0.3192	0.906504	0.898088
Gaussian Distribution Model Results						
Accuracy	**Precision**	**Recall**	**F1-score**	**RMSE**	**TPR**	**TNR**
.9921	0.6654	0.903	0.7336	0.0887	0.813	0.9929

In conclusion for experiment one, the best results for each model is represented in Figure 4.9. Gaussian distribution model achieved the highest TNR value among all models. But the full combined model of HMM, auto-encoder and K-means model reached the highest TPR value. So, the full combined model was considered as the best result in this experiment.

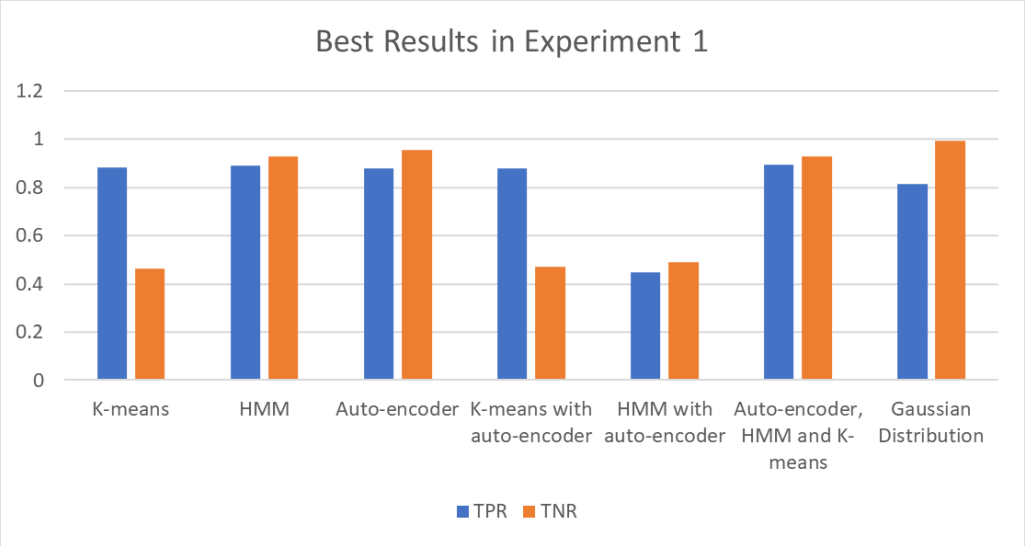

Figure 4.9: The Best Results in Experiment 1

4.4.2 Experiment 2 - Synthetic Dataset from a Financial Payment System

Experiment two was implemented on a synthetic dataset from a financial payment system. This dataset was generated using the BankSim payments simulator. BankSim is an agent-based simulator of bank payments based on a sample of aggregated transactional data provided by a bank in Spain. The main purpose of BankSim is the generation of synthetic data that can be used for fraud detection research. Statistical and Social Network Analysis (SNA) of relations between merchants and customers were used to develop and calibrate the model. The ultimate goal for BankSim is to be usable to model relevant scenarios that combine normal payments and injected known fraud signatures. The datasets generated by BankSim contain no personal information or disclosure of legal and private customer transactions. Therefore, it can be shared by academia, and others, to develop and research fraud detection methods. Synthetic data has the added benefit of being easier to acquire, faster and at less cost, for experimentation even for those that have access to their own data. BankSim generates data that approximates the relevant aspects of the real data. It has 180 steps (approximately six months) from BankSim with an average of three cards per step and performs about two fraudulent transactions per day. In total, it contains 594643 records, where 587443 are normal payments and 7200 are fraudulent transactions. It also

contains nine features which are time (step), Customer ID, Age, Gender, Zip Code, Merchant, Zip merchant, Category, Amount and Fraud. Table 4.12 describes some of the dataset characteristics.

Table 4.12: Dataset 2 Description

Dataset name	Synthetic dataset from a financial payment system
Dataset features number	10
Dataset observation number	594643
Dataset Date	------
Dataset place	Spain
Normal - Anomalous percentage	98.79 - 1.21%

To visualize the dataset, the histogram function in Python was applied on the dataset as shown in Figure 24:

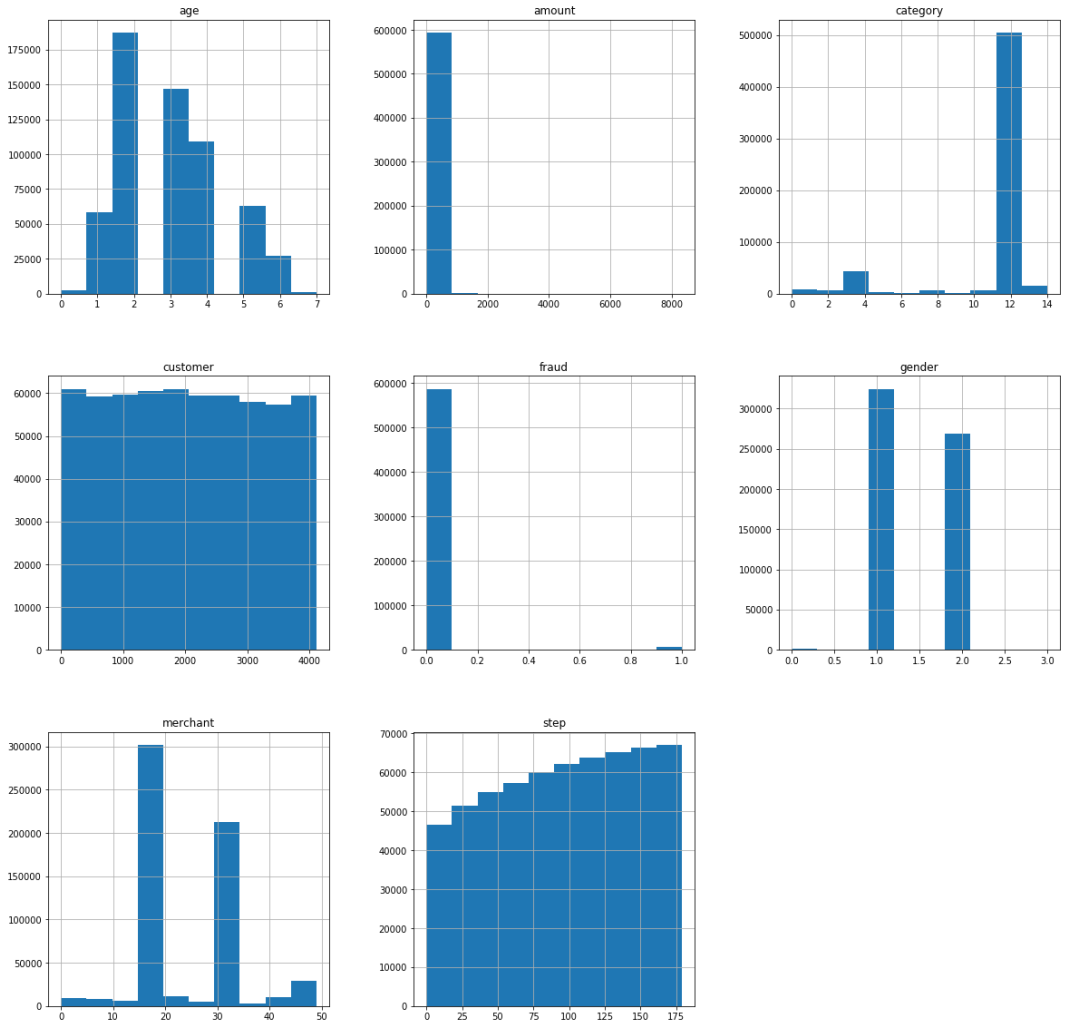

Figure 4.10: Features Histogram for Dataset 2

The dataset has some features that only have one value. For example, ‘Zip merchant’ has only one zip value ‘28007’ which will not affect the final predictions. These types of features were removed from the data before applying any model. There are several features that include letters that need to be categorized. For instance, ‘gender’ feature has two letter values; M for male and F for Female. The categorized process indicates the M as 1 and F as 2 in the dataset. There are no NAN values as all features are numbers. Finally, Feature importance was applied for applying PCA dimensional reduction. The features were sorted in terms of importance to the target using extra tree classifiers as shown in Figure 4.8. Additionally, a comparison was provided between data features in Appendix F.

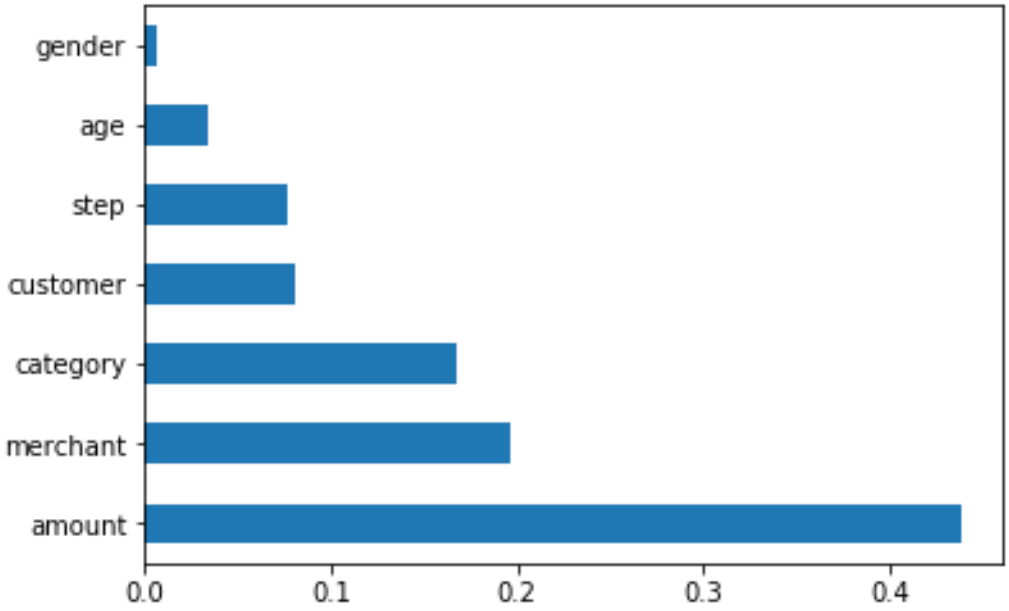

Figure 4.11: Feature Importance for Dataset 2

Default proposed models were applied between four assumptions for comparison between normalization and dimensional reduction. Table 4.6 shows the results with the four assumptions for Dataset 2. In the K-means model, the best result for TPR and TNR occurred by applying the fourth assumption of 74% and 57% respectively. So, it was chosen as the best assumption to be applied for tuning parameters.

In the HMM model, Table 4.6 shows that the best result for TPR of 100% was with the second assumption and has an acceptable TNR of 85%. The highest TNR was using

the third assumption of 88% with an acceptable TPR of 89%. So, the second assumption was chosen to be applied for tuning parameters because the TPR is much higher.

In the Auto-Encoder model, the best result for TPR of 100% was with the first and third assumptions. However, the best TNR of 99% was in the assumption of two and four. The TNR in the assumptions of one and three gives the highest TPR and has a very low TPR close to 0% which is not acceptable. So, the fourth assumption was chosen to be applied for tuning parameters because it has the highest TNR and acceptable TPR.

In the Gaussian Distribution model, the highest result for TPR and TNR was with assumption one of 58% and 99% respectively. So, the first assumption was chosen to be applied for tuning parameters. Finally, for more results such as F1 score and RMSE in this part refer to Appendix A, Appendix B, Appendix C, Appendix D.

Table 4.13: Results for Dataset 2 based on Four Assumptions

Models	Accuracy	TPR	TNR
Assumption 1: without normalization or dimensional reduction			
K-means	0.491382372	0.456944444	0.49243759
HMM	0.136783688	0	0.1409749
Auto-encoder	0.029730198	1	0
Gaussian	0.986208491	0.588055556	0.998408362
Assumption 2: with normalization only			
K-means	0.114015311	0.326111111	0.107516448
HMM	0.863216312	1	0.8590251
Auto-encoder	0.976215841	0.203611111	0.999889351
Gaussian	0.984276028	0.485833333	0.999548894
Assumption 3: with dimensional reduction only			
K-means	0.50835336	0.543055556	0.507290044
HMM	0.883441105	0.896388889	0.88304437
Auto-encoder	0.029738457	1	8.51E-06
Gaussian	0.984977991	0.520833333	0.999199925
Assumption 4: with Both normalization and dimensional reduction			
K-means	0.584470926	0.741944444	0.579645754
HMM	0.549992155	0.920277778	0.538646171
Auto-encoder	0.977677576	0.258333333	0.999719123
Gaussian	0.971830637	0.052777778	0.999991489

The best results have an outstanding accuracy in normal instances by 90% and good abnormal detection accuracy by 81% such as 3 and 200 in random states with 1 maximum iteration. Another group of results have less accuracy for normal and abnormal detection accuracy, for instance, 2, 4, 5, and 14 random states. Table 4.7 summarized all K-means results. Finally, for more results such as F1 score and RMSE in this part, refer to Appendix P.

Table 4.14: K-means Results for Dataset 2

Tuning Parameters		**Evaluations**		
Max Iter	**Random State**	**Accuracy**	**TPR**	**TNR**
1	0	0.324315173	0.201666667	0.328073266
10	0	0.411457688	0.251666667	0.416353871
1	42	0.317039533	0.614166667	0.307935211
10	42	0.413737003	0.255833333	0.418575356
1	1	0.426595314	0.370277778	0.428320949
10	1	0.416313621	0.258333333	0.421154321
1	2	0.570654642	0.664444444	0.567780814
10	2	0.583339527	0.741388889	0.57849671
1	3	0.900742429	0.811666667	0.903471814
10	3	0.579408534	0.739444444	0.574504847
1	4	0.563940573	0.652222222	0.56123552
10	4	0.58383503	0.741944444	0.578990374
1	5	0.688353195	0.783055556	0.685451404
10	5	0.588790064	0.749166667	0.583875937
1	13	0.098720776	0.189444444	0.095940897
10	13	0.419534392	0.260277778	0.4244142
1	14	0.508923189	0.716388889	0.502566198
10	14	0.581118021	0.740833333	0.576224157
1	90	0.447299094	0.411944444	0.448382402
10	90	0.418502094	0.259444444	0.423375805
1	91	0.48708801	0.271944444	0.493680259
10	91	0.418378218	0.259722222	0.423239622
1	200	0.901229674	0.809166667	0.904050592
10	200	0.411589822	0.251666667	0.416490054
1	250	0.346687147	0.410277778	0.344738656
10	250	0.413282792	0.255	0.418132761

The results in this model show better detections than K-means results in term of accuracy. It has higher accuracy for both normal and abnormal detection. The highest result for both normal and abnormal detection has a “diag” covariance type by 85% and 100%

respectively. The other results were varied with “spherical”, “tied” and “full” covariance type and some of them gave a satisfactory accuracy level for both, as shown in Table 4.9. Finally, for more results such as F1 score and RMSE in this part, refer to Appendix Q.

Table 4.15: HMM Results for Dataset 2

Tuning Parameters				**Evaluations**		
Covariance type	**N iter**	**algorithm**	**Tol**	**Accuracy**	**TPR**	**TNR**
Spherical	5k	viterbi	0.1	0.888544789	0.791944444	0.891504737
Diag	5k	viterbi	0.1	0.863216312	1	0.8590251
Tied	5k	viterbi	0.1	0.115006318	0.34	0.108112249
Full		viterbi		0.863216312	1	0.8590251
Spherical		viterbi		0.888544789	0.791944444	0.891504737
Diag		viterbi		0.863216312	1	0.8590251
Tied		viterbi		0.136783688	0	0.1409749
Spherical	5k	map	0.1	0.884993682	0.66	0.891887751
Diag	5k	map	0.1	0.888544789	0.791944444	0.891504737
Tied	5k	map	0.1	0.136783688	0	0.1409749
Full		map		0.115006318	0.34	0.108112249
Spherical		map		0.539941696	0.298611111	0.547336346
Diag		map		0.111455211	0.208055556	0.108495263
Tied		map		0.459727969	0.690277778	0.452663654
Spherical	5k	viterbi		0.115006318	0.34	0.108112249
Spherical	5	viterbi	0.1	0.136783688	0	0.1409749

The auto-encoder results were tuned using the following parameters: number of epochs, batch size, input dimension, encoding dimension, hidden dimension for layer 1, hidden dimension for layer 2, activation function, learning rate, and threshold. The best results were obtained by varying the threshold values, as shown in Table 4.10. The highest abnormal detection accuracy has two threshold values, but the normal detection accuracy has the lowest value, but it is an acceptable accuracy. Some other values have excellent accuracy for normal detection but an acceptable accuracy for abnormal detection. Overall, the auto-encoder has less accuracies in this Dataset from both previous models, as shown in Table 4.10. Finally, for more results such as F1 score and RMSE in this part refer to Appendix R.

Table 4.16: Auto-Encoder Model Results for Dataset 2

Tuning Parameters					**Evaluations**	
Encoding _dim	**Hidden _dim1**	**Hidden _dim2**	**Activation**	**Threshold**	**TPR**	**TNR**
18	10	6	tanh	4	0.3227778	0.9753424
18	10	6	tanh	4	0.3227778	0.9753424
32	16	8	tanh	4	0.3227778	0.9753424
10	5	2	tanh	4	0.3227778	0.9753424
5	2	1	tanh	4	0.3227778	0.9753424
5	3	1	tanh	4	0.3227778	0.9753424
50	20	10	tanh	4	0.335	0.9753424
5	2	1	sigmoid	4	0.335	0.9744997
5	2	1	hard_ sigmoid	4	0.335	0.9744997
5	2	1	exponentia l	4	0	1
5	2	1	linear	4	0.4766667	0.9449566
5	2	1	tanh	3	0.5805556	0.9371601
5	2	1	tanh	2	0.6986111	0.9281039
5	2	1	tanh	1	0.3863889	0.9596898
5	2	1	tanh	5	0	1
5	2	1	linear	4	0	1
5	2	1	tanh	4	0.3863889	0.9596898
5	2	1	tanh	4	0.3863889	0.9596898
5	2	1	tanh	4	0.3863889	0.9596898

The results of the rest of the models are shown in Table 4.11. Auto-Encoder with K-means model did not give more accuracy from the K-means model. However, there was good enhancement compared with the Auto-encoder results, especially TPR which is important in this research. The Auto-Encoder with HMM model gave much better results from Auto-Encoder results in term of TPR. The combination model between the three model (K-means, HMM, and Auto-Encoder) gave better results compared with Auto-Encoder results. Comparing these results with HMM and K-means results did not give a better result from the previous model. Finally, the Gaussian Distribution model reached the highest TNR with a non-acceptable TPR which shows that the Gaussian distribution model has a high ability to classify normal instances.

Table 4.17: Results of Four Models for Dataset 2

Evaluations						
K-means with Auto-encoder Model Results						
Accuracy	**Precision**	**Recall**	**F1-score**	**RMSE**	**TPR**	**TNR**
0.59696	0.52748720	0.732931	0.426856	0.63485619	0.8775	0.5883614
0.58921	0.52702032	0.730016	0.422835	0.64092804	0.879722	0.5803096
0.58754	0.52722214	0.731984	0.422245	0.64222811	0.885555	0.5784116
0.57303	0.52638416	0.726387	0.414667	0.65343339	0.88944	0.5633293
0.59517	0.52769252	0.734973	0.426225	0.6362595	0.88361	0.5863357
HMM with Auto-encoder Model Results						
Accuracy	**Precision**	**Recall**	**F1-score**	**RMSE**	**TPR**	**TNR**
0.481101	0.521607	0.685883	0.365147	0.720346	0.903611	0.468154466
0.3023974	0.515119	0.604968	0.256956	0.835226	0.926666	0.283269072
0.706027	0.485184	0.399104	0.420877	0.542193	0.072777	0.725429615
0.7092717	0.485091	0.399295	0.421778	0.5391922	0.06972	0.728868234
0.7014262	0.484796	0.395387	0.418917	0.5464190	0.07	0.72077386
0.7007325	0.484768	0.39502	0.418660	0.547053453	0.07	0.720058899
K-means, HMM, and Auto-encoder Model Results						
Accuracy	**Precision**	**Recall**	**F1-score**	**RMSE**	**TPR**	**TNR**
0.60572	0.477776	0.321294916	0.378560634	0.627917 3	0.0188888	0.623700
0.525398	0.499016	0.4914742	0.3591980	0.688913 1	0.8316666	0.516014
Gaussian Distribution Model Results						
Accuracy	**Precision**	**Recall**	**F1-score**	**RMSE**	**TPR**	**TNR**
0.971	0.9855	0.5119	0.516	0.170352 5	0.0238888	1

In conclusion of experiment two, the best results for each model are represented in Figure 4.12. The Gaussian distribution model achieved the highest TNR value among all

models with very low TPR which is not acceptable. But HMM model reached the highest TPR value. Therefore, the HMM model was considered the best result in this experiment.

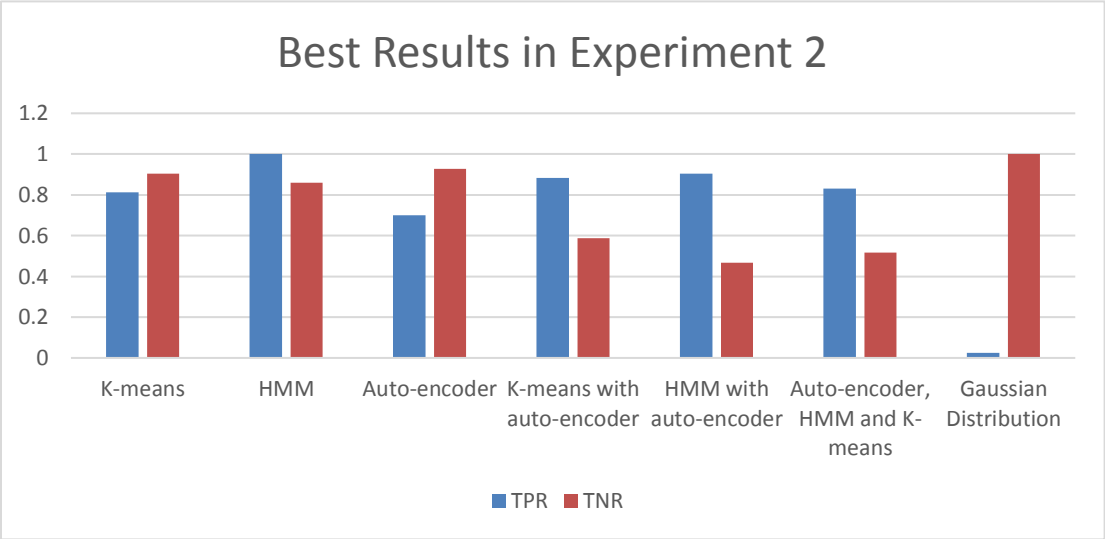

Figure 4.12: The Best Results in Experiment 2

4.4.3 Experiment 3 - German Credit Risk Dataset

Experiment three was implemented on a German Credit Risk dataset. This dataset contains data used to evaluate credit applications in Germany. It has 1000 entries with 24 numeric attributes (21 categorical, 3 real-valued). Each entry represents a person who takes a credit from a bank, and each person is classified as good or bad credit risks according to the set of attributes. There are no missing values. Seventy percent of the entries belong to a "Good" classification, while 30% are "Bad". Among the 24 attributes, 11 of them are bank account information such as saving amount, and credit history, while another 13 are personal information like age or whether they are a foreign worker or not. Table 3.1 describes some of the dataset characteristics. The attribute's description for this dataset as follows:

Attribute 1: Status of existing checking account

1 : ... < 0 DM, 2 : 0 <= ... < 200 DM

3 : ... >= 200 DM /salary assignments for at least 1 year, 4 : no checking account

Attribute 2: Duration in month

Attribute 3: Credit history

0 : no credits taken/all credits paid back duly

1 : all credits at this bank paid back duly

2 : existing credits paid back duly till now, 3 : delay in paying off in the past

4 : critical account/other credits existing (not at this bank)

Attribute 5: Credit amount

Attribute 6: Savings account/bonds

1 : ... < 100 DM, 2 : 100 <= ... < 500 DM, 3 : 500 <= ... < 1000 DM,

4 : .. >= 1000 DM, 5 : unknown/ no savings account

Attribute 7: (qualitative) Present employment since

1 : unemployed, 2 : ... < 1 year, 3 : 1 <= ... < 4 years,

4 : 4 <= ... < 7 years, 5 : .. >= 7 years

Attribute 9: Personal status and sex

1 : male : divorced/separated, 2 : female : divorced/separated/married

3 : male : single, 4 : male : married/widowed, 5 : female : single

Attribute 11: Present residence since

Attribute 12: (qualitative) Property

1 : real estate, 2 : if not A121 : building society savings agreement/ life insurance,

3 : if not A121/A122 : car or other, not in attribute 6, 4 : unknown / no property

Attribute 13: Age in years

Attribute 14: Other installment plans

1 : bank, 2 : store, 3 : none

Attribute 16: Number of existing credits at this bank

Attribute 18: Number of people being liable to provide maintenance for

Attribute 19: Telephone

1 : none, 2 : yes, registered under the customer's name

Attribute 20: foreign worker

1 : yes, 2 : no

Attribute 4_A40: Purpose

1 : car (new)

0 : car (used), furniture/equipment, radio/television, domestic appliances, repairs, education, (vacation - does not exist?), retraining, business, others

Attribute 4_A41: Purpose

1 : car (used)

0 : car (new), furniture/equipment, radio/television, domestic appliances, repairs, education, (vacation - does not exist?), retraining, business, others

Attribute 10_A101: Other debtors / guarantors

1 : none, 0 : co-applicant, 0 : guarantor

Attribute 10_A102: Other debtors / guarantors

0 : none, 1 : co-applicant, 0 : guarantor

Attribute 15_A151: Housing

1 : rent, 0 : own, 0 : for free

Attribute 15_A152: Housing

0 : rent, 1 : own, 0 : for free

Attribute 17_A171: Job

1 : unemployed/ unskilled - non-resident

0 : unskilled – resident, skilled employee / official, management/ self-employed/ highly qualified employee/ officer

Attribute 17_A171: Job

1 : unskilled - resident

0 : unemployed/ unskilled - non-resident, skilled employee / official, management/ self-employed/ highly qualified employee/ officer

Attribute 17_A171: Job

1 : skilled employee / official

0 : unemployed/ unskilled - non-resident, unskilled - resident, management/ self-employed/ highly qualified employee/ officer

Table 4.18: Dataset 3 Description

Dataset name	German Credit Risk dataset
Dataset features number	24
Dataset observation number	1000
Dataset Date	------
Dataset place	Germany

Normal - Anomalous percentage	70 - 30%

To visualize the dataset, the histogram function in Python was applied on the dataset as shown in Figure 4.10:

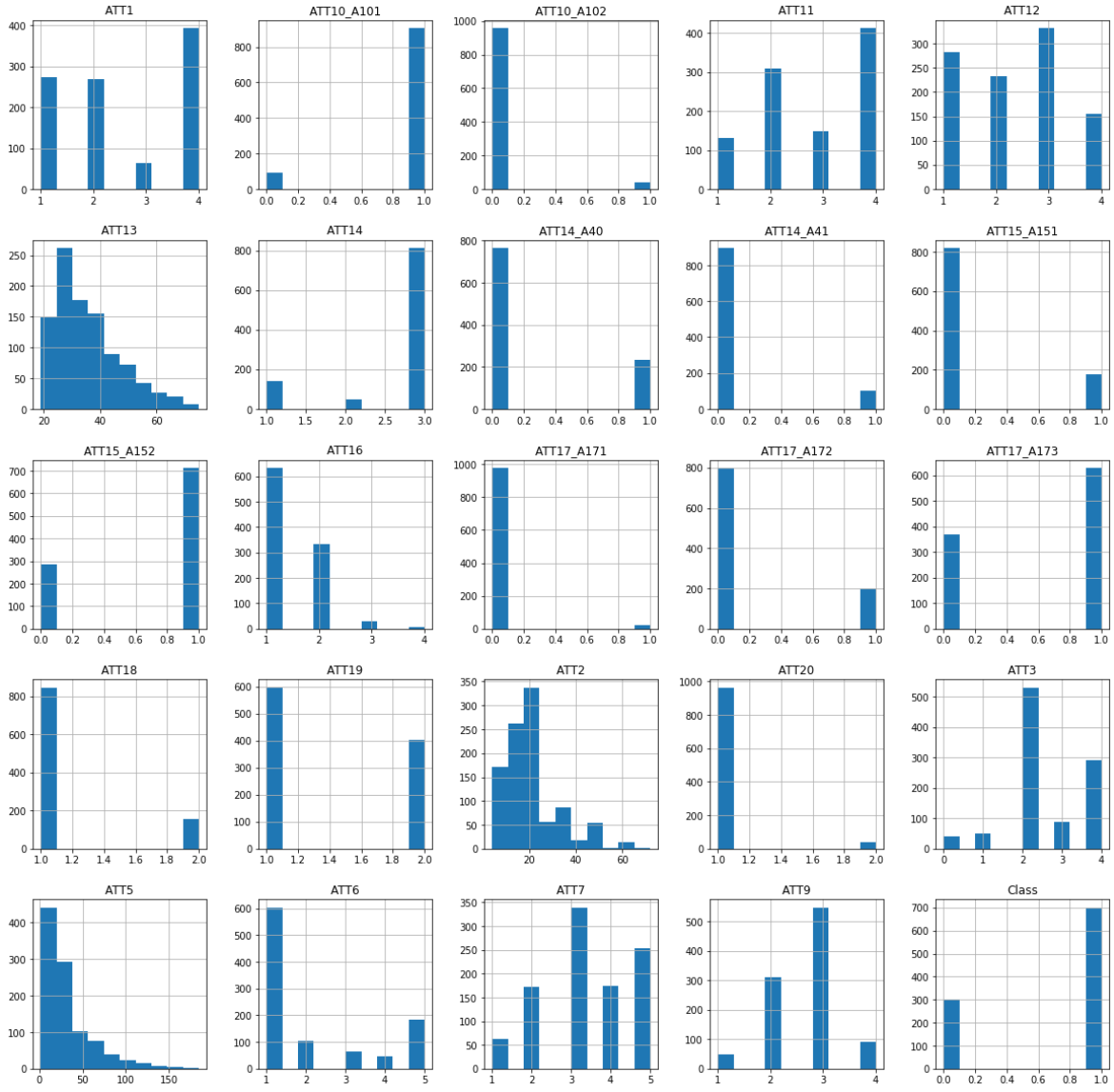

Figure 4.13: Features Histogram for Dataset 3

The Dataset has some features that need to be grouped. For example, the age feature has a range from 0 to 100. The new age feature is grouped into ten groups. The first group is indicated by 1 and gets the range from 0 to 10 and so on. Some features are extracted from the original features and delete the old ones. There are no NAN values as all features are numbers. Finally, Feature importance is applied for applying PCA dimensional

reduction. The features are sorted in term of importance to the target using extra tree classifiers as shown in Figure 4.8. Additionally, a comparison is provided between data features in Appendix F.

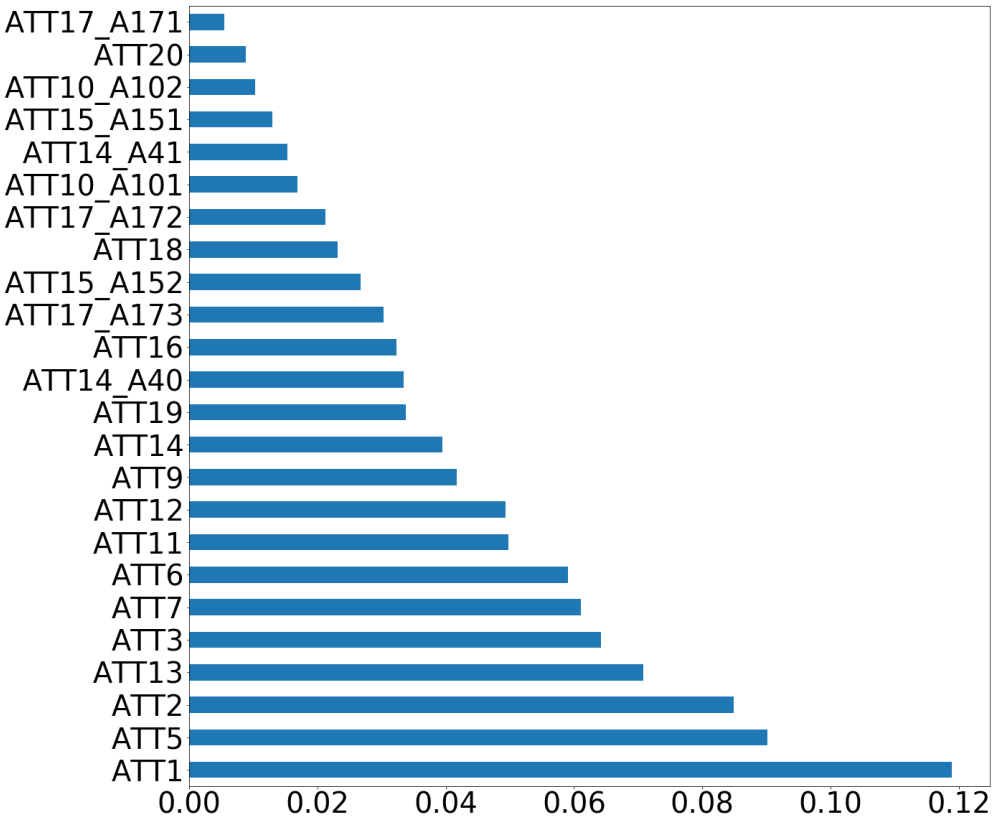

Figure 4.14: Feature Importance for Dataset 3

The default proposed models were applied between four assumptions for comparison between normalization and dimensional reduction. Table 4.6 shows the results with the four assumptions for Dataset 3. In the K-means model, the best result for TNR of 80% was by applying the first and third assumptions. But the second and the fourth assumptions gave the best TPR with a close result to the highest TNR, especially the fourth assumption. So, the fourth assumption was chosen as the best assumption to be applied for tuning parameters.

In the HMM model, Table 4.19 shows that the best result for TPR was with the fourth assumption of 38% and has an acceptable TNR of 70%. The highest TNR was by using the first assumption of 83% with very low TPR of 21%. So, the fourth assumption was chosen to be applied for tuning parameters because the TPR is highest.

In the Auto-Encoder model, the best result for TPR was with first assumption of 100%. In contrast, the best TNR of 100% was in assumption two. The TNR in the first assumption gave the highest TPR has 0% but has TNR which is not acceptable. The TPR in assumption two which gave the highest TNR has 0% TPR which is not acceptable. So, the fourth assumption was chosen to be applied for tuning parameters because it has the high TNR and is not 0 for TPR.

In the Gaussian Distribution model, the highest result for TPR was with assumption two but it is very low with a high TNR. However, the highest TNR was with assumptions three and four. So, the second assumption was chosen to be applied for tuning parameters. Finally, for more results such as F1 score and RMSE in this part refer to Appendix A, Appendix B, Appendix C, Appendix D.

Table 4.19: Results for Dataset 3 based on Four Assumptions

Models	Accuracy	TPR	TNR
Assumption 1: without normalization or dimensional reduction			
K-means	0.236585366	0.14	0.8
HMM	0.307317073	0.217142857	0.833333333
Auto-encoder	0.853658537	1	0
Gaussian	0.197560976	0.068571429	0.95
Assumption 2: with normalization only			
K-means	0.3	0.242857143	0.633333333
HMM	0.341463415	0.257142857	0.833333333
Auto-encoder	0.146341463	0	1
Gaussian	0.2	0.071428571	0.95
Assumption 3: with Dimensional reduction only			
K-means	0.236585366	0.14	0.8
HMM	0.353658537	0.291428571	0.716666667
Auto-encoder	0.83902439	0.982857143	0
Gaussian	0.146341463	0	1
Assumption 4: with Both normalization and Dimensional reduction			
K-means	0.302439024	0.242857143	0.65

HMM	0.426829268	0.38	0.7
Auto-encoder	0.173170732	0.034285714	0.983333333
Gaussian	0.148780488	0.002857143	1

Some results have an outstanding accuracy in the normal instances and unfortunate abnormal detection accuracy such as 0, 3 and 5 in random states. Another group of results are the opposite; unfortunate normal accuracy and excellent abnormal detection accuracy, for instance, 14, 42, 90, and 200 random states. Table 4.20 summarized all K-means results. Finally, for more results such as F1 score and RMSE in this part refer to Appendix S.

Table 4.20: K-means Results for Dataset 3

Tuning Parameters		**Evaluations**		
Max Iter	**Random State**	**Accuracy**	**TPR**	**TNR**
1	0	0.3	0.242857143	0.633333333
10	0	0.302439024	0.242857143	0.65
1	42	0.731707317	0.817142857	0.233333333
10	42	0.697560976	0.757142857	0.35
1	1	0.73902439	0.831428571	0.2
10	1	0.292682927	0.234285714	0.633333333
1	2	0.692682927	0.751428571	0.35
10	2	0.3	0.242857143	0.633333333
1	3	0.295121951	0.234285714	0.65
10	3	0.292682927	0.234285714	0.633333333
1	4	0.175609756	0.045714286	0.933333333
10	4	0.7	0.757142857	0.366666667
1	5	0.256097561	0.168571429	0.766666667
10	5	0.3	0.242857143	0.633333333
1	13	0.785365854	0.891428571	0.166666667
10	13	0.302439024	0.242857143	0.65
1	14	0.717073171	0.785714286	0.316666667
10	14	0.707317073	0.765714286	0.366666667
1	90	0.707317073	0.765714286	0.366666667
10	90	0.702439024	0.768571429	0.316666667
1	91	0.714634146	0.78	0.333333333
10	91	0.3	0.242857143	0.633333333
1	200	0.695121951	0.777142857	0.216666667
10	200	0.7	0.757142857	0.366666667
1	250	0.690243902	0.751428571	0.333333333
10	250	0.3	0.242857143	0.633333333

The results in this model showed better detections than K-means results in term of accuracy. It has higher accuracy for both normal and abnormal detection. The most acceptable result for both normal and abnormal detection has a "spherical" covariance type of 55% and 46% respectively. There were some results with an outstanding accuracy in the normal instances and unfortunate abnormal detection accuracy such as 'diag' with 'viterbi' in covariance type and algorithm respectively. Another group of results were the opposite; unfortunate normal accuracy and excellent abnormal detection accuracy, for instance, 'spherical' with 'viterbi' in covariance type and algorithm respectively. The other results were varied with "diag", "tied" and "full" covariance type and some of them gave a satisfactory accuracy level for both, as shown in Table 4.21. Finally, for more results such as F1 score and RMSE in this part, refer to Appendix T.

Table 4.21: HMM Results for Dataset 3

Tuning Parameters				**Evaluations**		
Covariance type	**N iter**	**algorithm**	**Tol**	**Accuracy**	**TPR**	**TNR**
Spherical	5k	viterbi	0.1	0.837142857	0.837142857	0.233333333
Diag	5k	viterbi	0.1	0.22	0.22	0.733333333
Tied	5k	viterbi	0.1	0.771428571	0.771428571	0.316666667
Full		viterbi		0.342857143	0.342857143	0.566666667
Spherical		viterbi		0.551428571	0.551428571	0.466666667
Diag		viterbi		0.714285714	0.714285714	0.283333333
Tied		viterbi		0.768571429	0.768571429	0.333333333
Spherical	5k	map	0.1	0.654285714	0.654285714	0.433333333
Diag	5k	map	0.1	0.845714286	0.845714286	0.233333333
Tied	5k	map	0.1	0.788571429	0.788571429	0.266666667
Full		map		0.771428571	0.771428571	0.316666667
Spherical		map		0.657142857	0.657142857	0.433333333
Diag		map		0.551428571	0.551428571	0.466666667
Tied		map		0.702857143	0.702857143	0.283333333
Spherical	5k	viterbi		0.231428571	0.231428571	0.666666667
Spherical	5	viterbi	0.1	0.345714286	0.345714286	0.566666667

The auto-encoder results are tuned using the following parameters: number of epochs, batch size, input dimension, encoding dimension, hidden dimension for layer 1, hidden dimension for layer 2, activation function, learning rate, and threshold. The best results are obtained with varying the threshold values, as shown in Table 4.22. The highest

abnormal detection accuracy has four threshold value with ‘tanh’ activation function of 66%, but the normal detection accuracy has very low value. Most of the other values have excellent accuracy for normal detection but unacceptable accuracy for abnormal detection. Finally, for more results such as F1 score and RMSE in this part refer to Appendix U.

Table 4.22: Auto-Encoder Model Results for Dataset 3

Tuning Parameters					**Evaluations**	
Encoding _dim	**Hidden _dim1**	**Hidden _dim2**	**Activation**	**threshold**	**TPR**	**TNR**
18	10	6	tanh	4	0.06	0.93333
18	10	6	tanh	4	0.048571429	0.98333
32	16	8	tanh	4	0.065714286	0.93333
10	5	2	tanh	4	0.062857143	0.93333
5	2	1	tanh	4	0.034285714	0.96667
5	3	1	tanh	4	0.077142857	0.91667
50	20	10	tanh	4	0.057142857	0.95
5	2	1	sigmoid	4	0.028571429	1
5	2	1	hard_ sigmoid	4	0.065714286	0.93333
5	2	1	exponential	4	0.065714286	0.93333
5	2	1	linear	4	0.054285714	0.96667
5	2	1	tanh	3	0.057142857	0.96667
5	2	1	tanh	2	0.057142857	0.96667
5	2	1	tanh	1	0.085714286	0.9
5	2	1	tanh	5	0.097142857	0.9
5	2	1	linear	4	0.22	0.78333
5	2	1	tanh	4	0.668571429	0.28333
5	2	1	tanh	4	0.031428571	0.9833
5	2	1	tanh	4	0.071428571	0.95

The results of the rest of the models were shown in Table 4.23. Auto-Encoder with K-means model gave more accuracy than the K-means and Auto-encoder model, especially TPR and TNR together which is important. The results in K-means and Auto-Encoder models separately were high for only one of the accuracies; TNR or TPR. However, in this model both TNR and TPR are increased in efficient values. Similarly, Auto-Encoder with HMM model gave much better results from Auto-Encoder and HMM models in terms of both accuracies together. The combination model between the three model (K-means, HMM, and Auto-Encoder) gave approximately the same results compared with the previous two models. Comparing these results with HMM, Auto-Encoder, and K-means results gave better results. Finally, the Gaussian Distribution model gave a high TPR with a non-acceptable TNR which shows that the Gaussian distribution model has high ability to classify the abnormal instances in this dataset.

Table 4.23: Results of Four Models for Dataset 3

Evaluations						
K-means with Auto-encoder Model Results						
Accuracy	**Precision**	**Recall**	**F1-score**	**RMSE**	**TPR**	**TNR**
0.453658537	0.465694683	0.431428571	0.383855732	0.431428571	0.739149148	0.462857143
0.451219512	0.46496747	0.43	0.382244812	0.43	0.740797197	0.46
0.446341463	0.463506261	0.427142857	0.379015847	0.427142857	0.744082345	0.454285714
0.495121951	0.502023376	0.504047619	0.427401345	0.504047619	0.710547571	0.491428571
0.509756098	0.506306371	0.512619048	0.437439843	0.512619048	0.700174194	0.508571429
HMM with Auto-encoder Model Results						
Accuracy	**Precision**	**Recall**	**F1-score**	**RMSE**	**TPR**	**TNR**
0.853658537	0.426829268	0.5	0.460526316	0.5	0.382546028	1
0.485365854	0.413048856	0.332619048	0.353800187	0.332619048	0.717380057	0.548571429
0.531707317	0.590995701	0.677380952	0.493944303	0.677380952	0.684319138	0.471428571
0.853658537	0.426829268	0.5	0.460526316	0.5	0.382546028	1
0.436585366	0.495106315	0.490476190	0.392023315	0.490476190	0.750609508	0.414285714
0.853658537	0.426829268	0.5	0.460526316	0.5	0.382546028	1
K-means, HMM, and Auto-encoder Model Results						
Accuracy	**Precision**	**Recall**	**F1-score**	**RMSE**	**TPR**	**TNR**
0.519512195	0.483336904	0.466666667	0.412675892	0.466666667	0.693172276	0.517142857
Gaussian Distribution Model Results						
Accuracy	**Precision**	**Recall**	**F1-score**	**RMSE**	**TPR**	**TNR**
0.148780488	0.573349633	0.501428571	0.130780773	0.501428571	0.922615582	0.002857143

In conclusion of experiment three, the best results for each model was represented in Figure 4.15. The Gaussian distribution model achieved the highest TPR value among all models with very low TNR which is not acceptable. But HMM with auto-encoder model

reached the highest TNR value. So, HMM with auto-encoder model was considered the best result in this experiment.

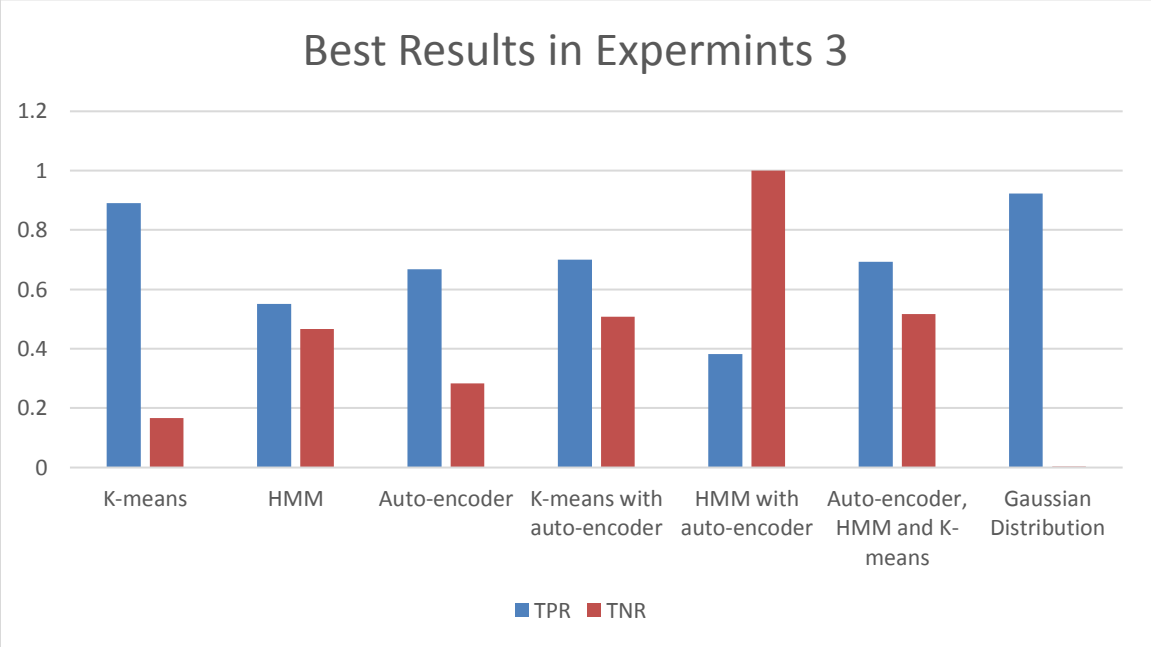

Figure 4.15: The Best Results in Experiment 3

4.4.4 Experiment 4 - Server Computers Dataset

Experiment four was implemented on server computers dataset. This Dataset has only two features. The features measure the through-put (mb/s) and latency (ms) of response of each server. While your servers were operating, you collected m = 307 examples of how they were behaving. Table 4.24 describes some of the dataset characteristics.

Table 4.24: Dataset 4 Description

Dataset name	Server computers dataset
Dataset features number	2
Dataset observation number	307
Dataset Date	------
Dataset place	------
Normal - Anomalous percentage	97.07 - 2.93%

To visualize the dataset, the histogram function in Python was applied on the dataset as shown in Figure 4.16:

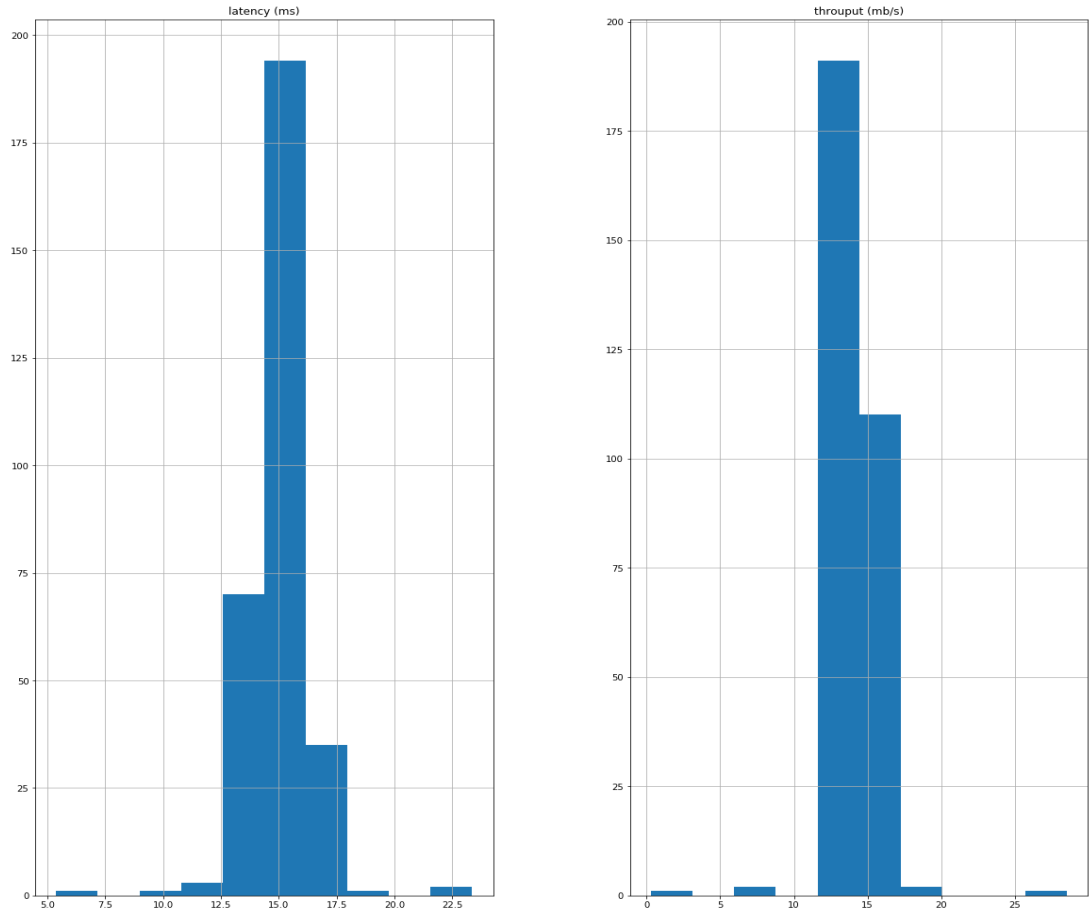

Figure 4.16: Features Histogram for Dataset 4

The Dataset is ready. There are no NAN values, all features are numbers. Finally, Feature importance was applied for applying PCA dimensional reduction. The features were sorted in terms of importance to the target using extra tree classifiers as shown in Figure 4.17.

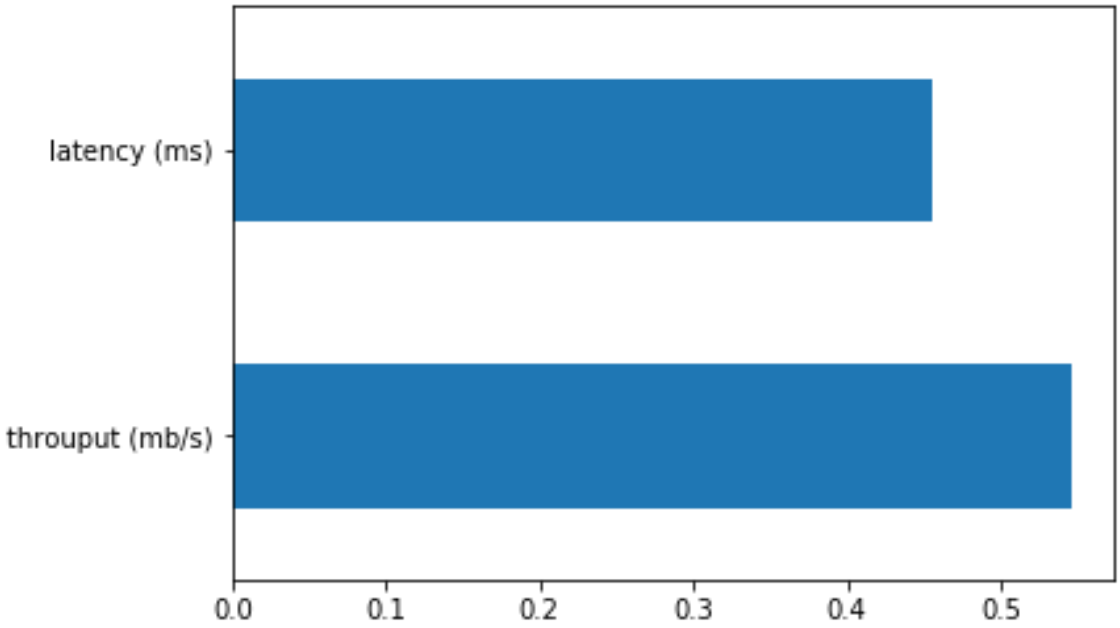

Figure 4.17: Feature Importance for Dataset 4

Default proposed models were applied between four assumptions for comparison between normalization and dimensional reduction. Table 4.25 shows the results with the four assumptions for Dataset 4. In the K-means model, the best result for TPR was achieved by applying the second assumption of 66% and has the highest TNR by approximately 50%. So, it was chosen as the best assumption to be applied for tuning parameters.

In the HMM model, Table 4.25 shows that the best result for TPR was with the second assumption of 77% and has a high TNR of 99%. The highest TNR was using the fourth assumption of 100% with an acceptable TPR of 66%. So, the first assumption was chosen to be applied for tuning parameters because the TPR is the highest.

In the Auto-Encoder model, the best result for TPR was with the first assumption of 100%. In contrast, the best TNR was in assumption fourth of 100%. The TNR in the first assumption gave the highest TPR but has 0% TNR which is not acceptable. The TPR in assumption four which gave the highest TNR has 66% TPR which is low. So, the second assumption was chosen to be applied for tuning parameters because it has the high TNR and high TPR. In the Gaussian Distribution model, the highest result for TNR was with

assumption one but it has very low TPR. Finally, for more results such as F1 score and RMSE in this part refer to Appendix A, Appendix B, Appendix C, Appendix D.

Table 4.25: Results for Dataset 4 based on Four Assumptions

Models	Accuracy	TPR	TNR
Assumption 1: without normalization or dimensional reduction			
K-means	0.488599349	0.555555556	0.486577181
HMM	0.990228013	0.777777778	0.996644295
Auto-encoder	0.029315961	1	0
Gaussian	0.977198697	0.222222222	1
Assumption 2: with normalization only			
K-means	0.5016	0.666666667	0.496644295
HMM	0.0098	0.222222222	0.003355705
Auto-encoder	0.9902	0.777777778	0.996644295
Gaussian	0.9772	0.222222222	1
Assumption 3: with Dimensional reduction only			
K-means	0.397394137	0.555555556	0.39261745
HMM	0.013029316	0.333333333	0.003355705
Auto-encoder	0.96742671	0.666666667	0.976510067
Gaussian	0.973941368	0.111111111	1
Assumption 4: with Both normalization and Dimensional reduction			
K-means	0.364820847	0.555555556	0.359060403
HMM	0.990228013	0.666666667	1
Auto-encoder	0.990228013	0.666666667	1
Gaussian	0.970684039	0	1

Some results have the highest accuracy in the normal instances and abnormal detection accuracy such as 2 and 5 in random states with one maximum iteration. Another group of results has less accuracies but is still acceptable, for instance, 1, 42 and 91 random states. Table 4.26 summarizes all K-means results. Finally, for more results such as F1 score and RMSE in this part refer to Appendix V.

Table 4.26: K-means Results for Dataset 4

Tuning Parameters		Evaluations		
Max Iter	**Random State**	**Accuracy**	**TPR**	**TNR**
1	0	0.5114	0.666666667	0.506711409
10	0	0.5016	0.666666667	0.496644295
1	42	0.4137	0.777777778	0.402684564
10	42	0.43	0.666666667	0.422818792
1	1	0.3322	0.444444444	0.32885906
10	1	0.4984	0.333333333	0.503355705
1	2	0.5505	0.666666667	0.546979866
10	2	0.5016	0.666666667	0.496644295
1	3	0.4495	0.333333333	0.453020134
10	3	0.4984	0.333333333	0.503355705
1	4	0.4886	0.333333333	0.493288591
10	4	0.4984	0.333333333	0.503355705
1	5	0.6352	0.555555556	0.637583893
10	5	0.5016	0.666666667	0.496644295
1	13	0.6189	0.333333333	0.627516779
10	13	0.5016	0.666666667	0.496644295
1	14	0.4267	0.444444444	0.426174497
10	14	0.4984	0.333333333	0.503355705
1	90	0.4365	0.444444444	0.436241611
10	90	0.4984	0.333333333	0.503355705
1	91	0.43	0.666666667	0.422818792
10	91	0.43	0.666666667	0.422818792
1	200	0.5147	0.444444444	0.516778523
10	200	0.57	0.333333333	0.577181208
1	250	0.645	0.222222222	0.657718121
10	250	0.5016	0.666666667	0.496644295

The results in this model showed better detections than K-means results in term of accuracy. It has higher accuracy for both normal and abnormal detection. The most acceptable result for both normal and abnormal detection has a “diag” and “viterbi” covariance type and algorithm of 100% and 77% respectively. There were some results with less accuracy in the normal instances and abnormal detection accuracy such as ‘full’ and ‘map’ in covariance type and algorithm respectively. The other results were varied with “spherical”, “tied” and “full” covariance type and some of them gave a satisfactory accuracy level for both, as shown in Table 4.27. Finally, for more results such as F1 score and RMSE in this part refer to Appendix W.

Table 4.27: HMM Results for Dataset 4

Tuning Parameters				**Evaluations**		
Covariance type	**N iter**	**algorithm**	**Tol**	**Accuracy**	**TPR**	**TNR**
Spherical	5k	viterbi	0.1	0.006514658	0.222222222	0
Diag	5k	viterbi	0.1	0.993485342	0.777777778	1
Tied	5k	viterbi	0.1	0.570032573	0.333333333	0.577181208
Full		viterbi		0.993485342	0.777777778	1
Spherical		viterbi		0.009771987	0.222222222	0.003355705
Diag		viterbi		0.990228013	0.777777778	0.996644295
Tied		viterbi		0.570032573	0.333333333	0.577181208
Spherical	5k	map	0.1	0.993485342	0.777777778	1
Diag	5k	map	0.1	0.006514658	0.222222222	0
Tied	5k	map	0.1	0.993485342	0.777777778	1
Full		map		0.557003257	0.333333333	0.563758389
Spherical		map		0.993485342	0.777777778	1
Diag		map		0.990228013	0.777777778	0.996644295
Tied		map		0.009771987	0.222222222	0.003355705
Spherical	5k	viterbi		0.442996743	0.666666667	0.436241611
Spherical	5	viterbi	0.1	0.006514658	0.222222222	0

The auto-encoder results were tuned using the following parameters: number of epochs, batch size, input dimension, encoding dimension, hidden dimension for layer 1, hidden dimension for layer 2, activation function, learning rate, and threshold. The best results were obtained by varying the threshold values, as shown in Table 4.28. The highest abnormal detection accuracy has a threshold value of 4, ‘tanh’ activation function, and the layers sequence is (10 – 5 – 2) by approximately 77%, and the normal detection accuracy has a high value of 100%. Most of the other values have excellent accuracy for normal detection but acceptable accuracy for abnormal detection. Finally, for more results such as F1 score and RMSE in this part refer to Appendix A, Appendix B, Appendix C, Appendix D. Finally, for more results such as F1 score and RMSE in this part refer to Appendix X.

Table 4.28: Auto-Encoder Model Results for Dataset 4

Tuning Parameters					**Evaluations**	
Encoding _dim	**Hidden _dim1**	**Hidden _dim2**	**activation**	**threshold**	**TPR**	**TNR**
18	10	6	tanh	4	0.7777778	0.9966443
18	10	6	tanh	4	0.44444444	1
32	16	8	tanh	4	0.77777778	1
10	5	2	tanh	4	0.77777778	1
5	2	1	tanh	4	0.77777778	0.993288591
5	3	1	tanh	4	0.77777778	0.996644295
50	20	10	tanh	4	0.55555556	1
5	2	1	sigmoid	4	0.44444444	1
5	2	1	hard_ sigmoid	4	0.77777778	0.996644295
5	2	1	exponential	4	0.77777778	0.996644295
5	2	1	linear	4	0.77777778	0.996644295
5	2	1	tanh	3	0.77777778	0.996644295
5	2	1	tanh	2	0.77777778	0.996644295
5	2	1	tanh	1	0.33333333	0.996644295
5	2	1	tanh	5	0.77777778	0.993288591
5	2	1	linear	4	0.77777778	0.976510067
5	2	1	tanh	4	0.77777778	0.865771812
5	2	1	tanh	4	0.77777778	0.996644295
5	2	1	tanh	4	0.33333333	0.996644295

The results of the rest of the models were shown in Table 4.29. Auto-Encoder with K-means model did not give more accuracy from the K-means and Auto-encoder model but still achieves an acceptable range of accuracy. Auto-Encoder with HMM model did not give much better results from Auto-Encoder and HMM models in terms of both accuracies

but still gives acceptable results. The combination model between the three model (K-means, HMM, and Auto-Encoder) gave approximately the same results compared with the previous two models. Comparing these results with HMM, Auto-Encoder, and K-means results did not have better results than the previous model. Finally, the Gaussian Distribution model gave an outstanding TNR with a non-acceptable TPR which shows that the Gaussian distribution model has a high ability to classify the normal instances in this dataset.

Table 4.29: Results of Four Models for Dataset 4

Evaluations						
K-means with Auto-encoder Model Results						
Accuracy	**Precision**	**Recall**	**F1-score**	**RMSE**	**TPR**	**TNR**
0.4267	0.4987	0.4892	0.3213	0.7571591	0.555555556	0.422818792
0.4691	0.4951	0.4571	0.3394	0.728659472	0.444444444	0.469798658
HMM with Auto-encoder Model Results						
Accuracy	**Precision**	**Recall**	**F1-score**	**RMSE**	**TPR**	**TNR**
0.491856678	0.502590234	0.522744221	0.356013339	0.7128	0.5556	0.4899
0.416938111	0.504483516	0.538031332	0.319829688	0.7636	0.6667	0.4094
K-means, HMM, and Auto-encoder Model Results						
Accuracy	**Precision**	**Recall**	**F1-score**	**RMSE**	**TPR**	**TNR**
0.5472	0.5059	0.5513	0.3841	0.672880918	0.555555556	0.546979866
Gaussian Distribution Model Results						
Accuracy	**Precision**	**Recall**	**F1-score**	**RMSE**	**TPR**	**TNR**
0.977198697	0.98852459	0.611111111	0.67601387	0.151001003	0.222222222	1

In conclusion of experiment four, the best results for each model was presented in Figure 4.18. Gaussian distribution, HMM and Auto-encoder models achieved the highest TPR value among the other models. But HMM and auto-encoder models reached the

highest TNR value. So, HMM or auto-encoder model was considered as the best result in this experiment.

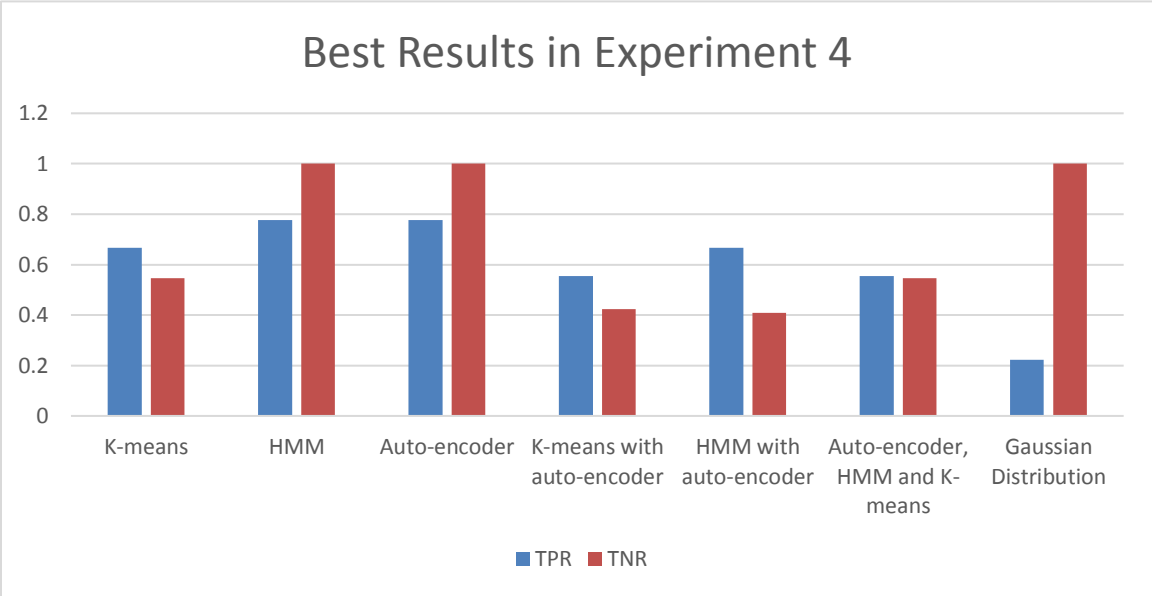

Figure 4.18: The Best Results in Experiment 4

4.4.5 Experiment 5 - High Dimensional Server Computers Dataset

Experiment five was implemented on a high dimensional server computers dataset. In this dataset, each example is described by 11 features, capturing many more properties of the computer servers. The features measure the through-put (mb/s) and latency (ms) of response of each server. While computers servers were operating, its collected m = 307 examples of how they were behaving. Table 4.30 describes some of the dataset characteristics.

Table 4.30: Dataset 5 Description

Dataset name	high dimensional server computers dataset
Dataset features number	11
Dataset observation number	1000
Dataset Date	------
Dataset place	------
Normal - Anomalous percentage	90.0 - 10.0%

To visualize the dataset, the histogram function in Python was applied on the dataset as shown in Figure 4.19:

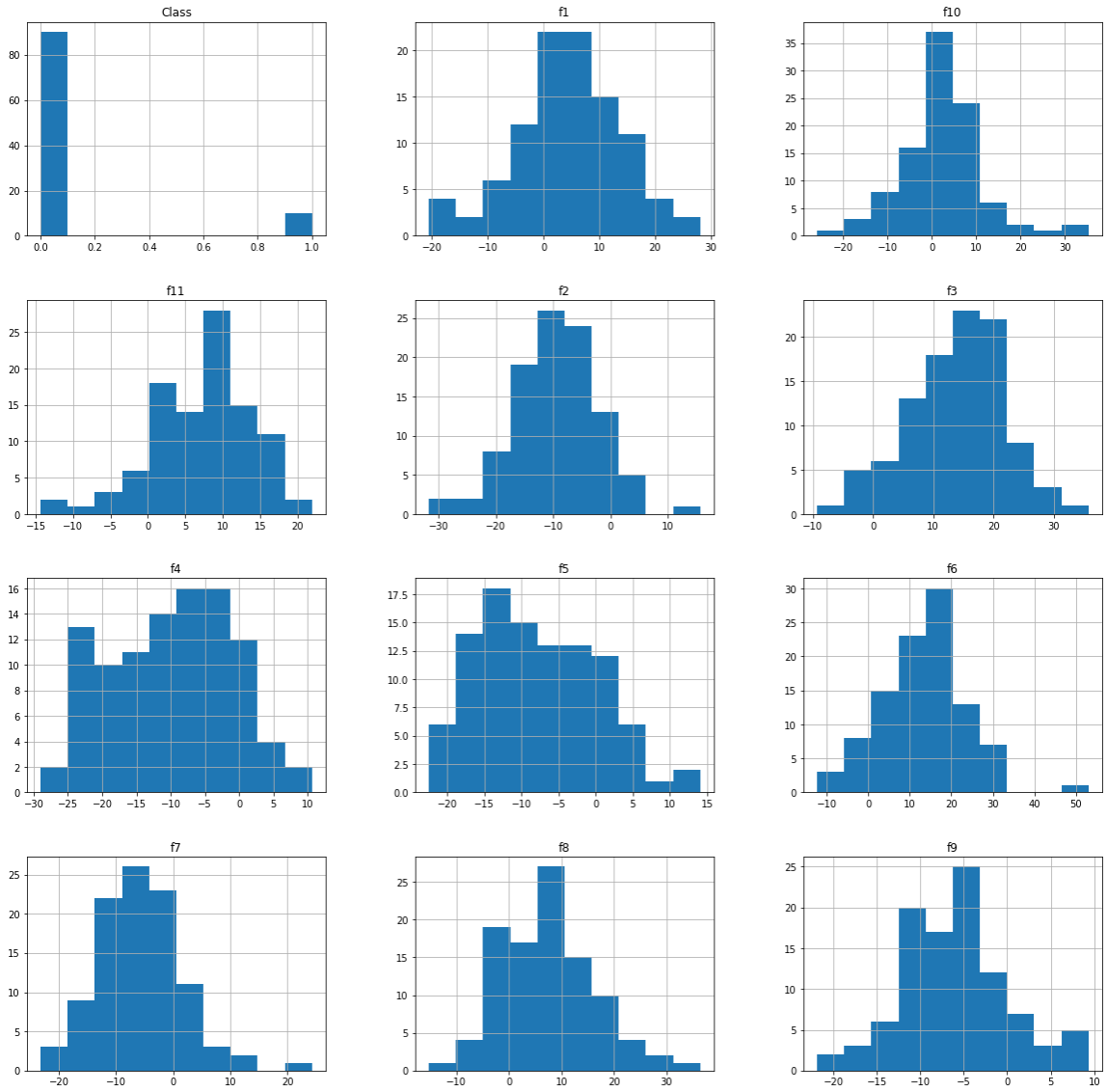

Figure 4.19: Features Histogram for Dataset 5

The Dataset is ready. There are no NAN values, all features are numbers. Finally, Feature importance was applied for applying PCA dimensional reduction. The features were sorted in term of importance to the target using extra tree classifier as shown in Figure 4.20. Additionally, a comparison was provided between data features in Appendix H.

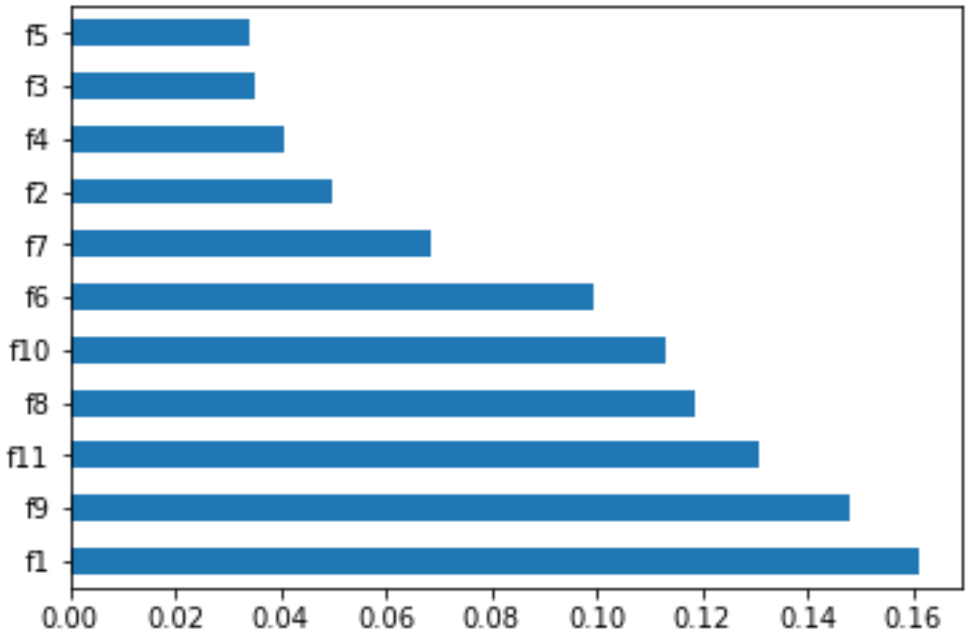

Figure 4.20: Feature Importance for Dataset 5

Default proposed models were applied between four assumptions for comparison between normalization and dimensional reduction. Table 4.31 shows the results with the four assumptions for Dataset 5. In the K-means model, the best result for TPR and TNR was by applying the second assumption among the four assumptions of 60% and 58% respectively. So, it was chosen as the best assumption to be applied for tuning parameters.

In the HMM model, Table 4.31 shows that the best result for TPR was 60% with the fourth assumption and has a highest TNR of 56%. So, the first assumption was chosen to be applied for tuning parameters because it has the highest values between all assumptions.

In the Auto-Encoder model, the best result for TPR was 100% with the third assumption. In contrast, the best TNR of 100% was in the first and second assumptions. The TNR in assumption three which gave the highest TPR has approximately 0% TNR which is not acceptable. The TPR in assumption one and two which gave the highest TNR has 0% TPR which is not acceptable. The fourth assumption has 40% TPR and 94% TNR which were a much better balance between the four assumptions. So, the fourth assumption

was chosen to be applied for tuning parameters because it has the suitable TNR and TPR. In the Gaussian Distribution model, the first assumption was chosen because it has the highest TPR and TNR results among all assumptions. Finally, for more results such as F1 score and RMSE in this part refer to Appendix A, Appendix B, Appendix C, Appendix D.

Table 4.31: Results for Dataset 5 based on Four Assumptions

Models	Accuracy	TPR	TNR
Assumption 1: without normalization or dimensional reduction			
K-means	0.52	0.2	0.555555556
HMM	0.54	0.4	0.555555556
Auto-encoder	0.1	0	1
Gaussian	0.92	0.2	1
Assumption 2: with normalization only			
K-means	0.59	0.6	0.588888889
HMM	0.51	0.5	0.511111111
Auto-encoder	0.9	0	1
Gaussian	0.9	0	1
Assumption 3: with Dimensional reduction only			
K-means	0.56	0.6	0.555555556
HMM	0.48	0.5	0.477777778
Auto-encoder	0.13	1	0.033333333
Gaussian	0.9	0	1
Assumption 4: with Both normalization and Dimensional reduction			
K-means	0.58	0.5	0.588888889
HMM	0.57	0.6	0.566666667
Auto-encoder	0.91	0.4	0.966666667
Gaussian	0.9	0	1

Some results have an outstanding accuracy in the normal instances and abnormal detection accuracy such as 1 in random states with ten maximum iteration. Another group of results has less accuracies but is still acceptable, for instance, 42 and 250 random states. Table 4.32 summarizes all K-means results. Finally, for more results such as F1 score and RMSE in this part refer to Appendix Y.

Table 4.32: K-means Results for Dataset 5

Tuning Parameters		Evaluations		
Max Iter	**Random State**	**Accuracy**	**TPR**	**TNR**
1	0	0.42	0.3	0.433333333
10	0	0.57	0.6	0.566666667
1	42	0.43	0.7	0.4
10	42	0.54	0.6	0.533333333
1	1	0.45	0.6	0.433333333
10	1	0.6	0.8	0.577777778
1	2	0.38	0.6	0.355555556
10	2	0.41	0.3	0.422222222
1	3	0.36	0.5	0.344444444
10	3	0.55	0.7	0.533333333
1	4	0.61	0.4	0.633333333
10	4	0.58	0.6	0.577777778
1	5	0.56	0.4	0.577777778
10	5	0.46	0.4	0.466666667
1	13	0.54	0.4	0.555555556
10	13	0.45	0.4	0.455555556
1	14	0.43	0.4	0.433333333
10	14	0.54	0.6	0.533333333
1	90	0.53	0.8	0.5
10	90	0.57	0.7	0.555555556
1	91	0.47	0.8	0.433333333
10	91	0.6	0.7	0.588888889
1	200	0.63	0.4	0.655555556
10	200	0.59	0.7	0.577777778
1	250	0.44	0.7	0.411111111
10	250	0.56	0.7	0.544444444

The results in this model showed better detections than K-means results in term of accuracy. It has higher accuracy for both normal and abnormal detection. The highest result for both normal and abnormal detection has a “spherical” covariance type of 60% and 61% respectively. There were some results with less accuracy in the normal instances and abnormal detection accuracy such as ‘tied’ with ‘map’ in covariance type and algorithm respectively. The other results were varied with “diag”, “tied” and “full” covariance type and some of them gave a satisfactory accuracy level for both, as shown in Table 4.33. Finally, for more results such as F1 score and RMSE in this part refer to Appendix Z.

Table 4.33: HMM Results for Dataset 5

Tuning Parameters				**Evaluations**		
Covariance type	**N iter**	**algorithm**	**Tol**	**Accuracy**	**TPR**	**TNR**
Spherical	5k	viterbi	0.1	0.59	0.6	0.588888889
Diag	5k	viterbi	0.1	0.57	0.6	0.566666667
Tied	5k	viterbi	0.1	0.56	0.6	0.555555556
Full	5k	map	0.1	0.56	0.6	0.555555556
Spherical		viterbi		0.61	0.6	0.611111111
Diag		viterbi		0.57	0.6	0.566666667
Tied		viterbi		0.53	0.5	0.533333333
Full	5k	viterbi	0.1	0.53	0.5	0.533333333
Spherical	5k	map	0.1	0.6	0.6	0.6
Diag	5k	map	0.1	0.56	0.6	0.555555556
Tied	5k	map	0.1	0.57	0.6	0.566666667
Full	5k	map	0.1	0.56	0.6	0.555555556
Spherical		map		0.61	0.6	0.611111111
Diag		map		0.56	0.6	0.555555556
Tied		map		0.54	0.5	0.544444444
Full		map		0.54	0.5	0.544444444
Spherical	5k	viterbi		0.61	0.6	0.611111111
Spherical	5	viterbi	0.1	0.59	0.6	0.588888889

The auto-encoder results were tuned using the following parameters: number of epochs, batch size, input dimension, encoding dimension, hidden dimension for layer 1, hidden dimension for layer 2, activation function, learning rate, and threshold. The best results were obtained by varying the threshold values, as shown in Table 4.34. The highest abnormal detection accuracy has four threshold values with ‘tanh’ activation function of approximately 80%, and the normal detection accuracy has a value 55%. Most of the other values have excellent accuracy for normal detection but unacceptable accuracy for abnormal detection. Finally, for more results such as F1 score and RMSE in this part refer to Appendix AA.

Table 4.34: Auto-Encoder Model Results for Dataset 5

Tuning Parameters					**Evaluations**	
Encoding _dim	**Hidden _dim1**	**Hidden _dim2**	**activation**	**threshold**	**TPR**	**TNR**
18	10	6	tanh	4	0.2	1
18	10	6	tanh	4	0.1	1
32	16	8	tanh	4	0.2	1
10	5	2	tanh	4	0.4	0.966666667
5	2	1	tanh	4	0.4	0.955555556
5	3	1	tanh	4	0.4	0.966666667
50	20	10	tanh	4	0.2	1
5	2	1	sigmoid	4	0.2	1
5	2	1	hard_sigmoid	4	0.3	0.988888889
5	2	1	exponential	4	0.4	0.955555556
5	2	1	linear	4	0.4	0.933333333
5	2	1	tanh	3	0.4	0.933333333
5	2	1	tanh	2	1	0
5	2	1	tanh	1	0.5	0.955555556
5	2	1	tanh	5	0.5	0.888888889
5	2	1	linear	4	0.5	0.855555556
5	2	1	tanh	4	0.8	0.555555556
5	2	1	tanh	4	0.4	0.966666667
5	2	1	tanh	4	0.4	0.955555556

The results of the rest of the models were shown in Table 4.35. Auto-Encoder with K-means model did not give more accuracy compared to the K-means and Auto-encoder model but still has an acceptable range of accuracy. Auto-Encoder with HMM model gave a small increase in results compared to the Auto-Encoder model. The combination model between the three model (K-means, HMM, and Auto-Encoder) gave approximately the same results compared with the previous two models. Comparing these results with HMM, Auto-Encoder, and K-means results did not have better results than the previous model. Finally, the Gaussian Distribution model gave an outstanding TNR with a non-acceptable TPR which shows that the Gaussian distribution model has a high ability to classify the normal instances in this dataset.

Table 4.35: Results of Four Models for Dataset 5

Evaluations						
K-means with Auto-encoder Model Results						
Accuracy	**Precision**	**Recall**	**F1-score**	**RMSE**	**TPR**	**TNR**
0.47	0.51010101	0.527777778	0.396011396	0.728010989	0.6	0.455555556
0.47	0.51010101	0.527777778	0.396011396	0.728010989	0.6	0.455555556
HMM with Auto-encoder Model Results						
Accuracy	**Precision**	**Recall**	**F1-score**	**RMSE**	**TPR**	**TNR**
0.52	0.536057692	0.6	0.438990182	0.6928	0.7	0.5
0.56	0.528180354	0.577777778	0.454365079	0.6633	0.6	0.5556
0.700732519	0.484767631	0.39502945	0.418660852	0.39502945	0.547053453	0.07
K-means, HMM, and Auto-encoder Model Results						
Accuracy	**Precision**	**Recall**	**F1-score**	**RMSE**	**TPR**	**TNR**
0.55	0.542016807	0.616666667	0.4590696	0.670820393	0.7	0.533333333
Gaussian Distribution Model Results						
Accuracy	**Precision**	**Recall**	**F1-score**	**RMSE**	**TPR**	**TNR**
0.92	0.959183673	0.6	0.645390071	0.282842712	0.2	1

In conclusion of experiment five, the best results for each model was represented in Figure 4.21. The Gaussian distribution model achieved the highest TNR value among the other models with very low TPR. However, the K-means model reached the highest TPR value with acceptable TNR. So, the K-means model was considered as the best result in this experiment.

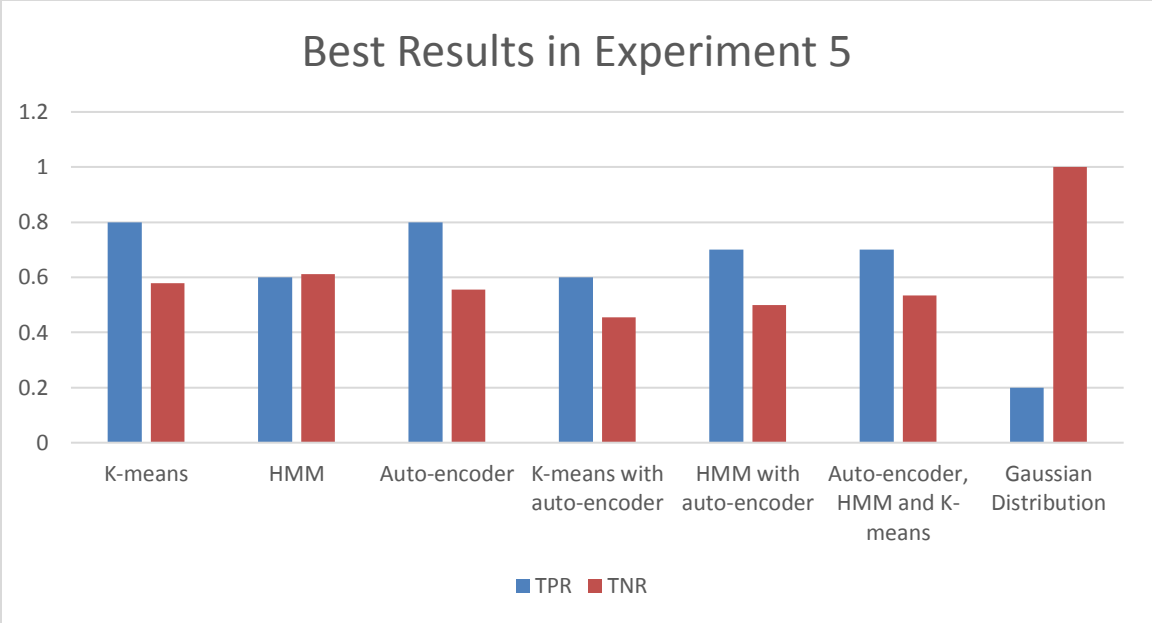

Figure 4.21: The Best Results in Experiment 5

4.4.6 Experiment 6 - Transmission History Dataset

Experiment six was implemented on a transmission history dataset (conn250K). There are 256670 records total, each of which is with 4 fields that will be described. "record ID" - the unique identifier for each connection record. "duration_" - This feature denotes the number of seconds (rounded) of the connection. For example, a connection for 0.17s or 0.3s would be indicated with a "0" in this field. "src_bytes" - This field represents the number of data bytes transferred from the source to the destination (i.e., the amount of out-going bytes from the host). "dst_bytes" - This feature represents the number of data bytes transferred from the destination to the source (i.e., the amount of bytes received by the host). Table 4.36 describes some of the dataset characteristics.

Table 4.36: Dataset 6 Description

Dataset name	Transmission History Dataset dataset
Dataset features number	4
Dataset observation number	256670
Dataset Date	------
Dataset place	------
Normal - Anomalous percentage	99.62 - 0.38%

To visualize the dataset, the histogram function in Python was applied on the dataset as shown in Figure 4.22:

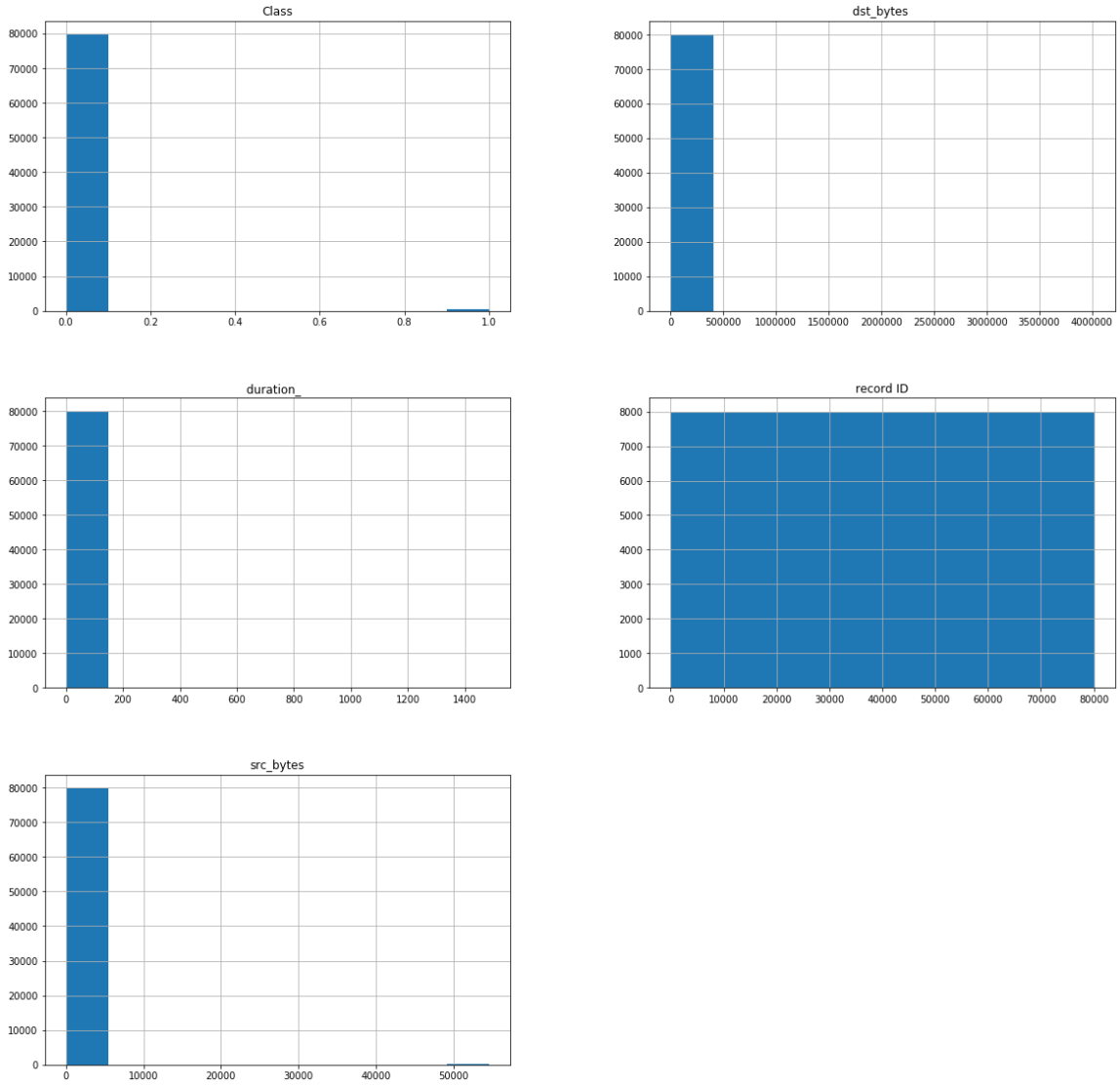

Figure 4.22: Features Histogram for Dataset 6

The Dataset is ready. There are no NAN values, all features are numbers. Finally, Feature importance was applied for applying PCA dimensional reduction. The features were sorted in term of importance to the target using extra tree classifiers as shown in Figure 4.23. Additionally, a comparison is provided between data features in Appendix I.

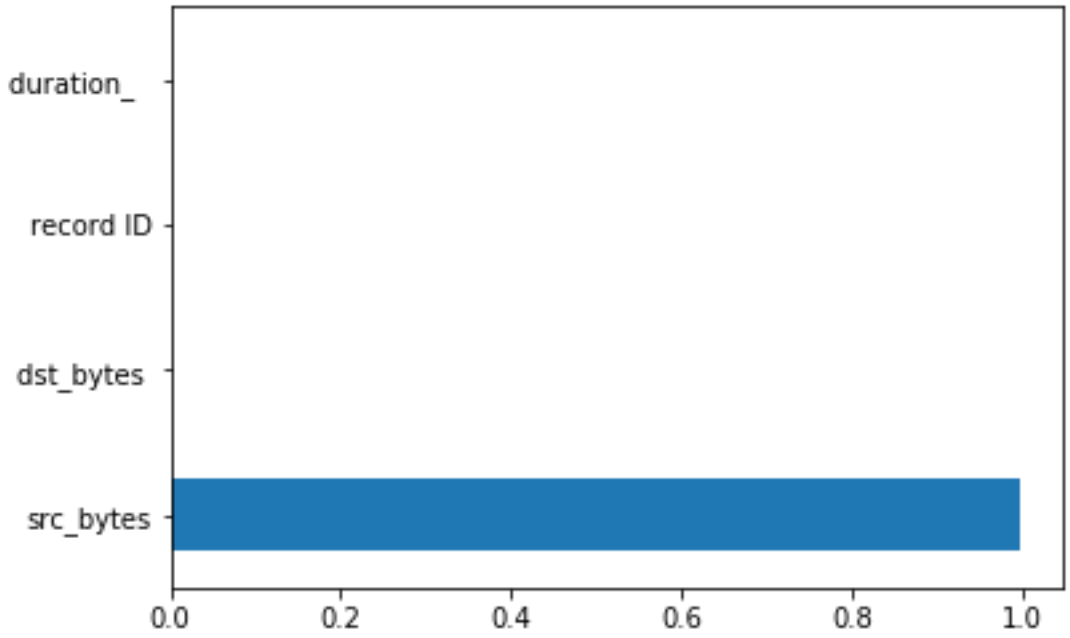

Figure 4.23: Feature Importance for Dataset 6

Default proposed models were applied between four assumptions for comparison between normalization and dimensional reduction. Table 4.37 shows the results with the four assumptions for Dataset 6. In the K-means model, the best result for TNR of 99% was by applying the fourth assumption among the four assumptions. However, the TPR for the fourth assumption is not acceptable. The highest TPR of 52% was with assumption three and a TNR of 50%. So, the third assumption was chosen as the best assumption to be applied for tuning parameters.

In the HMM model, Table 4.37 shows that the first assumption got the highest values in TNR and TPR among all assumptions. So, the first assumption was chosen to be applied for tuning parameters.

In the Auto-Encoder model, the best result for TPR and TNR was by applying the fourth assumption which yielded 100% and 99% respectively. So, the fourth assumption was chosen to be applied for tuning parameters because it has the highest TNR and TPR values. In the Gaussian Distribution model, the second assumption was chosen because it has the highest TPR and TNR results among all assumptions. Finally, for more results such

as F1 score and RMSE in this part refer to Appendix A, Appendix B, Appendix C, Appendix D.

Table 4.37: Results for Dataset 6 based on Four Assumptions

Models	Accuracy	TPR	TNR
Assumption 1: without normalization or dimensional reduction			
K-means	0.498664015	0.474025974	0.498902064
HMM	0.986205182	1	0.986071899
Auto-encoder	0.009569378	1	0
Gaussian	0.998197974	1	0.998180563
Assumption 2: with normalization only			
K-means	0.498726154	0.474025974	0.498964803
HMM	0.980736966	1	0.98055085
Auto-encoder	0.995898838	1	0.995859213
Gaussian	0.998322252	1	0.998306042
Assumption 3: with Dimensional Reduction only			
K-means	0.501149568	0.525974026	0.500909718
HMM	0.498726154	0.474025974	0.498964803
Auto-encoder	0.009755794	1	0.000188218
Gaussian	0.990430622	0	1
Assumption 4: with Both normalization and Dimensional Reduction			
K-means	0.988317902	0	0.997866867
HMM	0.979121357	1	0.978919631
Auto-encoder	0.996644504	1	0.996612084
Gaussian	0.998011558	0.993506494	0.998055085

Most results have an outstanding accuracy around 97% in the normal instances and 0% for abnormal detection accuracy such as 0, 1, and 42 in random states. Two results have 100% abnormal accuracy but close to 0% normal accuracy with 250 random states. Overall, K-means model did not work well in this dataset. Table 4.38 summarized all K-means results. Finally, for more results such as F1 score and RMSE in this part refer to Appendix BB.

Table 4.38: K-means Results for Dataset 6

Tuning Parameters		Evaluations		
Max Iter	**Random State**	**Accuracy**	**TPR**	**TNR**
1	0	0.988317902	0	0.997866867
10	0	0.988317902	0	0.997866867
1	42	0.988255763	0	0.997804128
10	42	0.988317902	0	0.997866867
1	1	0.988317902	0	0.997866867
10	1	0.988317902	0	0.997866867
1	2	0.988690735	0	0.998243303
10	2	0.988317902	0	0.997866867
1	3	0.988317902	0	0.997866867
10	3	0.988317902	0	0.997866867
1	4	0.988317902	0	0.997866867
10	4	0.988317902	0	0.997866867
1	5	0.988317902	0	0.997866867
10	5	0.988317902	0	0.997866867
1	13	0.988317902	0	0.997866867
10	13	0.988317902	0	0.997866867
1	14	0.988317902	0	0.997866867
10	14	0.446405269	0	0.450718364
1	90	0.988317902	0	0.997866867
10	90	0.988317902	0	0.997866867
1	91	0.988317902	0	0.997866867
10	91	0.988317902	0	0.997866867
1	200	0.988317902	0	0.997866867
10	200	0.988317902	0	0.997866867
1	250	0.012241347	1	0.002697785
10	250	0.011682098	1	0.002133133

The results in this model show better detections than K-means results in term of accuracy. It has higher accuracy for both normal and abnormal detection together. The highest result for both normal and abnormal detection has a “diag” covariance type of 98% and 100% respectively. There are some results with less accuracy than the normal accuracy such as ‘spherical’ with ‘viterbi’ in covariance type and algorithm respectively. The other results are varied with “diag”, “tied” and “full” covariance type and some of them give a satisfactory accuracy level for both, as shown in Table 4.39. Finally, for more results such as F1 score and RMSE in this part refer to Appendix CC.

Table 4.39: HMM Results for Dataset 6

Tuning Parameters				**Evaluations**		
Covariance type	**N iter**	**algorithm**	**Tol**	**Accuracy**	**TPR**	**TNR**
Spherical	5k	viterbi	0.1	0.402100292	1	0.396323483
Diag	5k	viterbi	0.1	0.986267321	1	0.986134638
Tied	5k	viterbi	0.1	0.50369726	0.681818182	0.501976285
Full	5k	map	0.1	0.98632946	1	0.986197378
Spherical		viterbi		0.40166532	1	0.395884309
Diag		viterbi		0.986267321	1	0.986134638
Tied		viterbi		0.49630274	0.31818182	0.498023715
Full	5k	viterbi	0.1	0.98632946	1	0.986197378
Spherical	5k	map	0.1	0.402100292	1	0.396323483
Diag	5k	map	0.1	0.986267321	1	0.986134638
Tied	5k	map	0.1	0.503759398	0.68181818	0.502039024
Full	5k	map	0.1	0.98632946	1	0.986197378
Spherical		map		0.401603182	1	0.39582157
Diag		map		0.986267321	1	0.986134638
Tied		map		0.496240602	0.318181818	0.497960976
Full		map		0.98632946	1	0.986197378
Spherical	5k	viterbi		0.40166532	1	0.395884309
Spherical	5	viterbi	0.1	0.44478966	0.99350649	0.439488048

The auto-encoder results were tuned using the following parameters: number of epochs, batch size, input dimension, encoding dimension, hidden dimension for layer 1, hidden dimension for layer 2, activation function, learning rate, and threshold. The best results were obtained by varying the threshold values, as shown in Table 4.40. Most of the results gave the highest normal and abnormal detection accuracies by 99% and 100%. Only two results have 0% abnormal accuracy and 100% normal detection but unacceptable accuracy for abnormal detection. Finally, for more results such as F1 score and RMSE in this part refer to Appendix DD.

Table 4.40: Auto-Encoder Model Results for Dataset 6

Tuning Parameters					**Evaluations**	
Encoding _dim	**Hidden _dim1**	**Hidden _dim2**	**activation**	**threshold**	**TPR**	**TNR**
18	10	6	tanh	4	1	0.996235648
18	10	6	tanh	4	1	0.996235648
32	16	8	tanh	4	1	0.996235648
10	5	2	tanh	4	1	0.996235648
5	2	1	tanh	4	1	0.996235648
5	3	1	tanh	4	1	0.996235648
50	20	10	tanh	4	1	0.996235648
5	2	1	sigmoid	4	1	0.996235648
5	2	1	hard_ sigmoid	4	1	0.996235648
5	2	1	exponential	4	1	0.996047431
5	2	1	linear	4	1	0.994039777
5	2	1	tanh	3	1	0.993851559
5	2	1	tanh	2	1	0.993851559
5	2	1	tanh	1	0	1
5	2	1	tanh	5	1	0.995921952
5	2	1	linear	4	1	0.995545517
5	2	1	tanh	4	1	0.993475124
5	2	1	tanh	4	1	0.996612084
5	2	1	tanh	4	0	1

The results of the rest models were shown in Table 4.41. Auto-Encoder with K-means model gave more accuracy than the K-means but the results of Auto-Encoder was better. Auto-Encoder with HMM model gave the same accuracy level because the results of the two models separately were very high. The combination model between the three model (K-means, HMM, and Auto-Encoder) gave approximately the same results compared with the previous two models. Comparing these results with HMM, Auto-Encoder, and K-means results did not have better results than the previous model. Finally, the Gaussian Distribution model gave an outstanding TNR and TPR which shows that the Gaussian distribution model has a high ability to classify the normal and abnormal instances in this dataset.

Table 4.41: Results of Four Models for Dataset 6

Evaluations						
K-means with Auto-encoder Model Results						
Accuracy	**Precision**	**Recall**	**F1-score**	**RMSE**	**TPR**	**TNR**
0.580562978	0.511029343	0.785039839	0.387382042	0.647639577	0.993506494	0.576573185
0.590070217	0.511281502	0.789839388	0.391701119	0.640257591	0.993506494	0.586172282
0.449450071	0.489698583	0.230111676	0.310179725	0.741990518	0.006493506	0.453729845
0.448517989	0.489555073	0.226425748	0.309639226	0.742618348	0	0.452851496
0.592369353	0.511468573	0.794215446	0.392870733	0.638459589	1	0.588430893
HMM with Auto-encoder Model Results						
Accuracy	**Precision**	**Recall**	**F1-score**	**RMSE**	**TPR**	**TNR**
0.962778848	0.601831591	0.977994228	0.659486271	0.192927842	0.993506494	0.962481962
0.981296216	0.668842203	0.987342368	0.747293483	0.136761779	0.993506494	0.981178242
0.974523085	0.636088803	0.983923082	0.707173101	0.159614897	0.993506494	0.97433967
0.961660349	0.599870298	0.980644959	0.656617196	0.195805135	1	0.961289918
0.972534642	0.629194631	0.986134638	0.698303177	0.165726756	1	0.972269277
K-means, HMM, and Auto-encoder Model Results						
Accuracy	**Precision**	**Recall**	**F1-score**	**RMSE**	**TPR**	**TNR**
0.9636	0.6041	0.9816	0.6629	0.1908	1	0.9632
Gaussian Distribution Model Results						
Accuracy	**Precision**	**Recall**	**F1-score**	**RMSE**	**TPR**	**TNR**
0.998	0.9157	0.9958	0.9522	0.044591952	0.993506494	0.998055085

In conclusion of experiment five, the best results for each model was presented in Figure 4.24. Most of the models in this data got the highest TPR with 100%. But Auto-encoder model reached the highest TPR value with high TNR. So, Auto-encoder model was considered as the best result in this experiment.

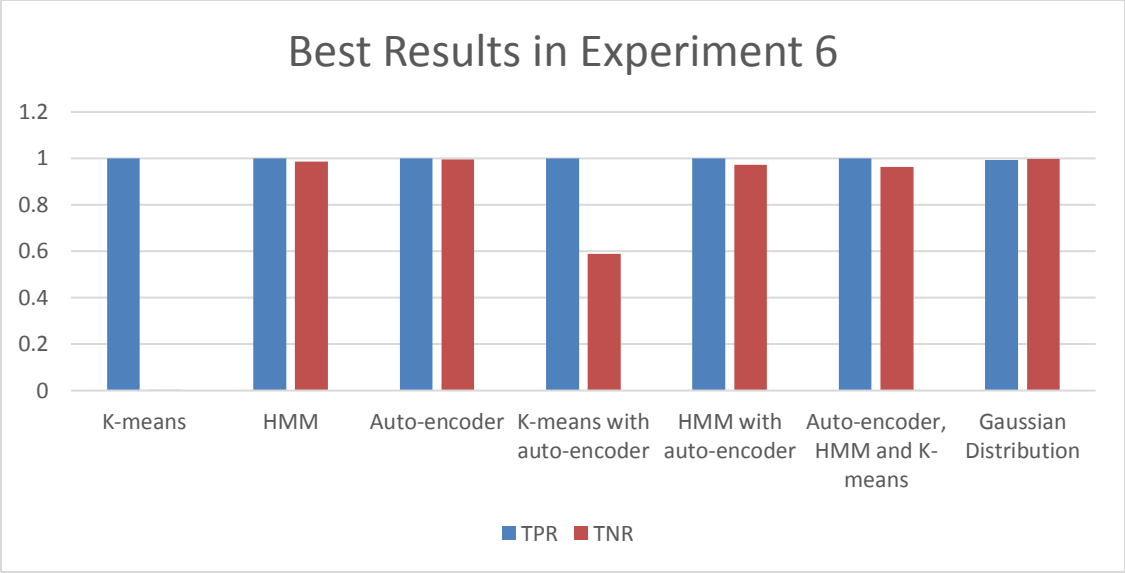

Figure 4.24: The Best Results in Experiment 6

4.4.7 Experiment 7 - Porto Seguro's Safe Driver Prediction Dataset

Experiment seven was implemented on a Porto Seguro's Safe Driver Prediction dataset. There are 595212 observations in total and each observation is described by 59 features. Table 4.42 describes some of the dataset characteristics which is provided by Porto Seguro. Porto Seguro is one of the Brazil's largest auto and homeowner insurance companies. Inaccuracies in car insurance company's claim predictions raise the cost of insurance for good drivers and reduce the price for bad ones. Features that belong to similar groupings are tagged as such in the feature names (e.g., ind, reg, car, calc). In addition, feature names include the postfix bin to indicate binary features and cat to indicate categorical features. Features without these designations are either continuous or ordinal. Values of -1 indicate that the feature was missing from the observation. The target columns signify whether or not a claim was filed for that policy holder.

Table 4.42: Dataset 7 Description

Dataset name	Porto Seguro's Safe Driver Prediction dataset
Dataset features number	59
Dataset observation number	595212
Dataset Date	------
Dataset place	Brazil
Normal - Anomalous percentage	96.36 - 3.64%

To visualize the dataset, the histogram function in Python was applied on the dataset, four features is shown in Figure 4.25 as sample:

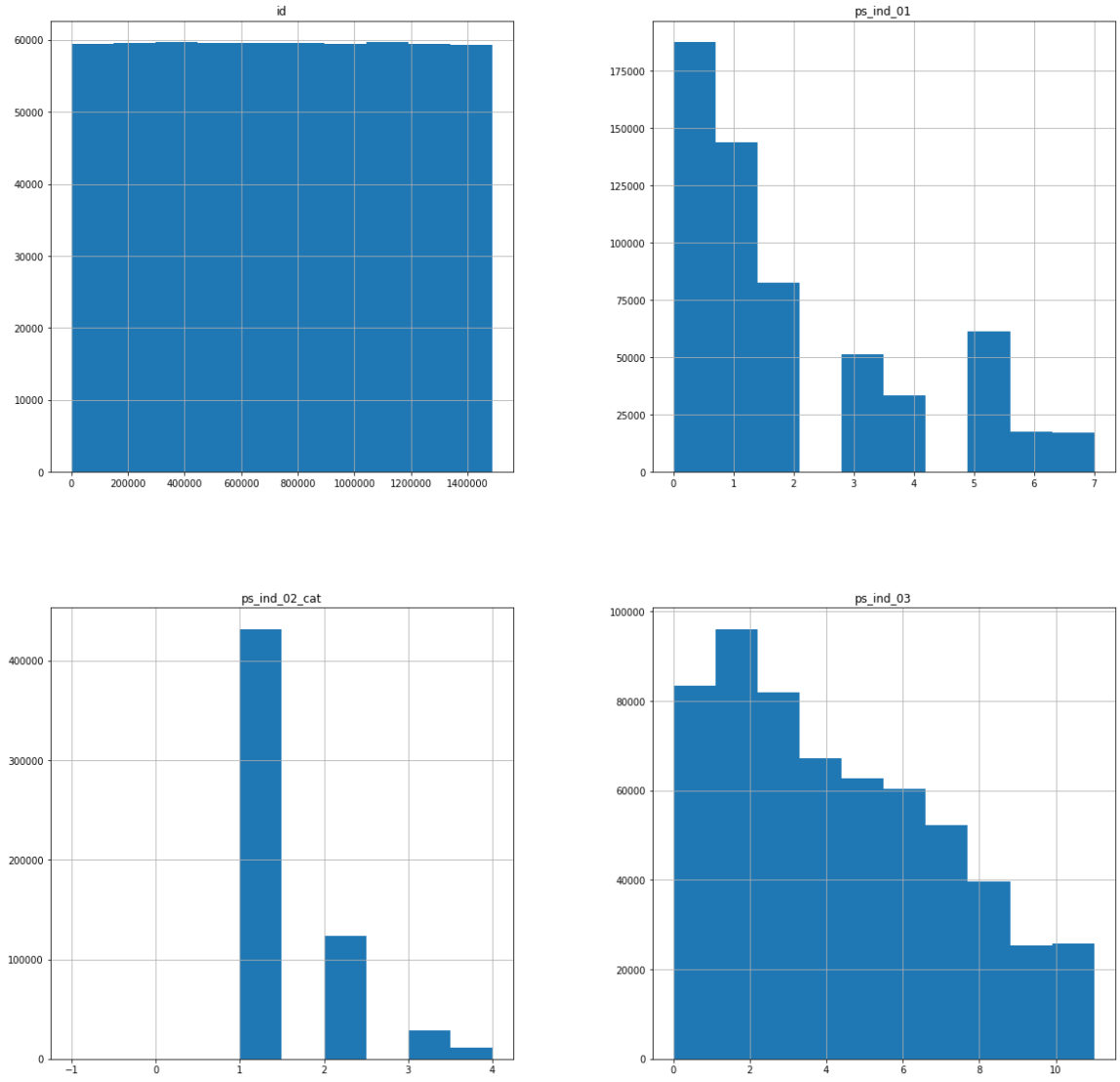

Figure 4.25: Features Histogram for Dataset 7

The Dataset is ready. There are no NAN values, all features are numbers. Finally, Feature importance was applied for applying PCA dimensional reduction. The features were sorted in term of importance to the target using extra tree classifiers as shown in Figure 4.26. Additionally, a comparison was provided between data features in Appendix J.

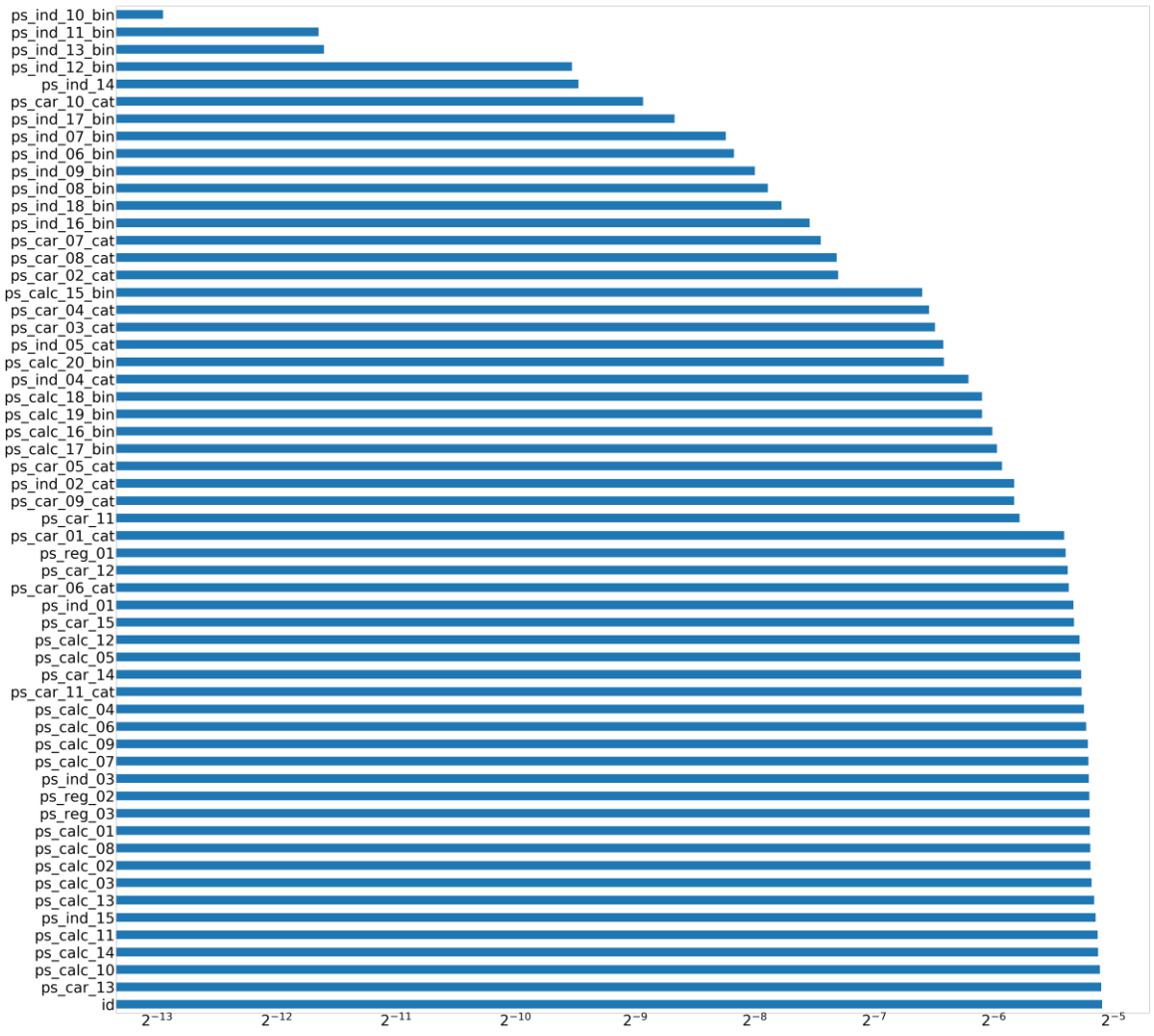

Figure 4.26: Feature Importance for Dataset 7

Default proposed models were applied between four assumptions for comparison between normalization and dimensional reduction. Table 4.43 shows the results with the four assumptions for Dataset 7. In the K-means model, the best result for TNR of 70% was achieved by applying the fourth assumption among the four assumptions. However, the TPR for the fourth assumption was very low. The highest TPR of 58% was with assumption two with very low TNR. The third assumption has a suitable TPR and TNR of 50% for both accuracies. So, the third assumption was chosen as the best assumption to be applied for tuning parameters.

In the HMM model, Table 4.43 shows that the first assumption got the highest TPR value among all assumptions but has a very low TNR. However, the best TNR value was with the fourth assumption with low TPR. So, the fourth assumption was chosen to be applied for tuning parameters.

In the Auto-Encoder model, the best result for TPR of 100% was achieved by applying the first and third assumptions but the TNR was 0% in these assumptions. The best result for TNR of 99% was with applying the second and fourth assumptions. But the TPR was very low in these assumptions. So, the fourth assumption was chosen to be applied for tuning parameters because it has the highest TNR and has better TPR compared to the second assumption. In the Gaussian Distribution model, the fourth assumption was chosen because it has the most suitable results for TPR and TNR whereas the other assumptions have 0% for one of the rates. Finally, for more results such as F1 score and RMSE in this part refer to Appendix A, Appendix B, Appendix C, Appendix D.

Table 4.43: Results for Dataset 7 based on Four Assumptions

Models	Accuracy	TPR	TNR
Assumption 1: without normalization or dimensional reduction			
K-means	0.500171245	0.505393196	0.499677431
HMM	0.101297481	0.975938047	0.018586972
Auto-encoder	0.08639517	1	0
Gaussian	0.91360483	0	1
Assumption 2: with normalization only			
K-means	0.313291013	0.582465198	0.287836518
HMM	0.852792889	0.118189361	0.922260776
Auto-encoder	0.911454309	0.003872038	0.997279955
Gaussian	0.08639517	1	0
Assumption 3: with Dimensional reduction only			
K-means	0.500824366	0.504747857	0.500453341
HMM	0.183200452	0.812943671	0.123648696
Auto-encoder	0.08639517	1	0
Gaussian	0.91360483	0	1
Assumption 4: with Both normalization and Dimensional reduction			
K-means	0.682782296	0.424080391	0.707246478
HMM	0.865743801	0.101687102	0.937996931
Auto-encoder	0.908085161	0.012077072	0.992816292
Gaussian	0.894353689	0.035401494	0.975580625

The best result for TNR has an outstanding accuracy around 81% in the normal instances and 29% for abnormal detection accuracy with 4 in random states and 1 iteration. The result for ten iterations with 4 random state gave 71% for TNR and 40% for abnormal accuracy. The highest result in the abnormal accuracy was 65% with 23% for the normal accuracy with 14 random state. Table 4.44 summarizes all K-means results. Finally, for more results such as F1 score and RMSE in this part refer Appendix EE.

Table 4.44: K-means Results for Dataset 7

Tuning Parameters		**Evaluations**		
Max Iter	**Random State**	**Accuracy**	**TPR**	**TNR**
1	0	0.382466089	0.496174057	0.371713279
10	0	0.335823689	0.547524661	0.315804157
1	42	0.420418794	0.443440583	0.418241735
10	42	0.311076774	0.584401217	0.285229809
1	1	0.484719357	0.385636582	0.494089134
10	1	0.65041298	0.469622937	0.667509416
1	2	0.539016017	0.357979165	0.556135793
10	2	0.326106522	0.564580068	0.303555238
1	3	0.396165702	0.478104545	0.388417143
10	3	0.332223559	0.554807781	0.31117485
1	4	0.766270281	0.294551489	0.810878435
10	4	0.697732396	0.40278418	0.725624215
1	5	0.341423007	0.571402231	0.31967499
10	5	0.330965106	0.556467226	0.309640466
1	13	0.312040525	0.601456624	0.284671851
10	13	0.33175363	0.554992164	0.310643046
1	14	0.274294908	0.658615285	0.237951597
10	14	0.309619199	0.586798193	0.283407728
1	90	0.378324346	0.518207799	0.365096248
10	90	0.322761268	0.56670047	0.299693123
1	91	0.438172535	0.42666175	0.439261055
10	91	0.33792642	0.546326173	0.318219068
1	200	0.56252041	0.55277957	0.563441554
10	200	0.662933788	0.453120679	0.682774794
1	250	0.337647649	0.54807781	0.317748291
10	250	0.334541342	0.550843551	0.314086693

The results in this model showed better detections than K-means results in term of higher accuracy. It has higher accuracy for both normal and abnormal detection but not together. The highest result for normal detection of 96% has a “spherical” covariance type

with close to 0% in the abnormal accuracy. The highest result for abnormal detection of 95% has a “spherical” covariance type with close to 0% in the normal accuracy. There were “tied” results with less accuracy in the normal accuracy with 30% for abnormal accuracy. The other results were varied with “diag”, “tied” and “full” covariance type and some of them gave a satisfactory accuracy level for both, as shown in Table 4.45. Finally, for more results such as F1 score and RMSE in this part refer to Appendix FF.

Table 4.45: HMM Results for Dataset 7

Tuning Parameters				**Evaluations**		
Covariance type	**N iter**	**algorithm**	**Tol**	**Accuracy**	**TPR**	**TNR**
Spherical	5k	viterbi	0.1	0.107494166	0.956301281	0.027226601
Diag	5k	viterbi	0.1	0.147207111	0.881810639	0.077739224
Spherical		viterbi		0.887145463	0.060569743	0.965310713
Diag		viterbi		0.852792889	0.118189361	0.922260776
Tied		viterbi		0.769448272	0.258504656	0.817765727
Spherical	5k	map	0.1	0.107494166	0.956301281	0.027226601
Diag	5k	map	0.1	0.852792889	0.118189361	0.922260776
Tied	5k	map	0.1	0.743403079	0.308380197	0.78454108
Spherical		map		0.112464258	0.94025998	0.034183638
Diag		map		0.852792889	0.118189361	0.922260776
Tied		map		0.778719405	0.231953536	0.830424397
Spherical	5k	viterbi		0.107494166	0.956301281	0.027226601
Spherical	5	viterbi	0.1	0.834163009	0.174702683	0.896524969

The auto-encoder results were tuned using the following parameters: number of epochs, batch size, input dimension, encoding dimension, hidden dimension for layer 1, hidden dimension for layer 2, activation function, learning rate, and threshold. The best results were obtained by varying the threshold values, as shown in Table 4.46. Most of the results gave the highest normal detection accuracies of 99%. But the abnormal detection accuracy was very low. Only one result has 63% abnormal accuracy and 45% for normal detection. Overall, Auto-Encoder did not detect well in this dataset. Finally, for more results such as F1 score and RMSE in this part refer to Appendix GG.

Table 4.46: Auto-Encoder Model Results for Dataset 7

Tuning Parameters					**Evaluations**	
Encoding _dim	**Hidden _dim1**	**Hidden _dim2**	**activation**	**threshold**	**TPR**	**TNR**
18	10	6	tanh	4	0.012353646	0.992380388
18	10	6	tanh	4	0.01594911	0.99013112
32	16	8	tanh	4	0.014105283	0.991386525
10	5	2	tanh	4	0.019636766	0.987332613
5	2	1	tanh	4	0.02055868	0.986417213
5	3	1	tanh	4	0.020927445	0.986966453
50	20	10	tanh	4	0.010325436	0.993539894
5	2	1	sigmoid	4	0.011339541	0.99319117
5	2	1	hard_ sigmoid	4	0.020835254	0.98647824
5	2	1	exponential	4	0.02120402	0.986356186
5	2	1	linear	4	0.021572785	0.986146952
5	2	1	tanh	3	0.024154144	0.984760776
5	2	1	tanh	2	0.024154144	0.984760776
5	2	1	tanh	1	0.019728957	0.986652601
5	2	1	tanh	5	0.031068498	0.978483749
5	2	1	linear	4	0.078915829	0.954430534
5	2	1	tanh	4	0.631787591	0.451196122
5	2	1	tanh	4	0.012538029	0.993208607
5	2	1	tanh	4	0.019728957	0.986696192

The results of the rest of the models were shown in Table 4.47. Auto-Encoder with K-means model gave more suitable accuracy than Auto-encoder but the results of K-means was better. Auto-Encoder with HMM model did not give better accuracy level than auto-encoder or HMM. The combination model between the three model (K-means, HMM, and Auto-Encoder) gave approximately the same results compare with the previous two models. Comparing these results with HMM, Auto-Encoder, and K-means results did not give better results than the previous model. Finally, the Gaussian Distribution model gave an outstanding TPR which shows that the Gaussian distribution model has a high ability to classify the normal instances in this dataset.

Table 4.47: Results of Four Models for Dataset 7

Evaluations						
K-means with Auto-encoder Model Results						
Accuracy	**Precision**	**Recall**	**F1-score**	**RMSE**	**TPR**	**TNR**
0.41743196	0.514699067	0.544024038	0.361078294	0.763261449	0.697059095	0.39098898
0.417423995	0.514683561	0.543977942	0.361066243	0.763266667	0.696966903	0.39098898
0.582456532	0.48529634	0.455956673	0.419271086	0.646176035	0.303033097	0.608880248
0.584503509	0.485300376	0.456075268	0.420035829	0.644590173	0.300820503	0.611330032
0.415408878	0.51469908	0.543918519	0.359858519	0.764585589	0.699271688	0.38856 5351
HMM with Auto-encoder Model Results						
Accuracy	**Precision**	**Recall**	**F1-score**	**RMSE**	**TPR**	**TNR**
0.749950219	0.494001307	0.488148324	0.480344563	0.500049778	0.171660367	0.804636281
0.75099362	0.494158862	0.488510676	0.480765832	0.499005391	0.17119941	0.805821942
0.756577009	0.494378249	0.489229115	0.482276829	0.493379156	0.166036692	0.812421537
0.252972895	0.508293275	0.516498221	0.244043885	0.864307298	0.835069604	0.197926838
0.246139019	0.507914 54	0.515262359	0.238386519	0.868251681	0.840601088	0.18992363
K-means, HMM, and Auto-encoder Model Results						
Accuracy	**Precision**	**Recall**	**F1-score**	**RMSE**	**TPR**	**TNR**
0.358109454	0.516990961	0.546115809	0.324049093	0.8012	0.7734	0.3188
Gaussian Distribution Model Results						
Accuracy	**Precision**	**Recall**	**F1-score**	**RMSE**	**TPR**	**TNR**
0.8944	0.5175	0.5055	0.4994	0.325032785	0.035401494	0.975580625

In conclusion of experiment seven, the best results for each model was represented in Figure 4.27. The Gaussian distribution model got the highest TNR of 97% with very low TPR. But the most balanced accuracy for both TPR and TNR was using K-means with

Auto-encoder model. So, K-means with Auto-encoder model was considered as the best result in this experiment.

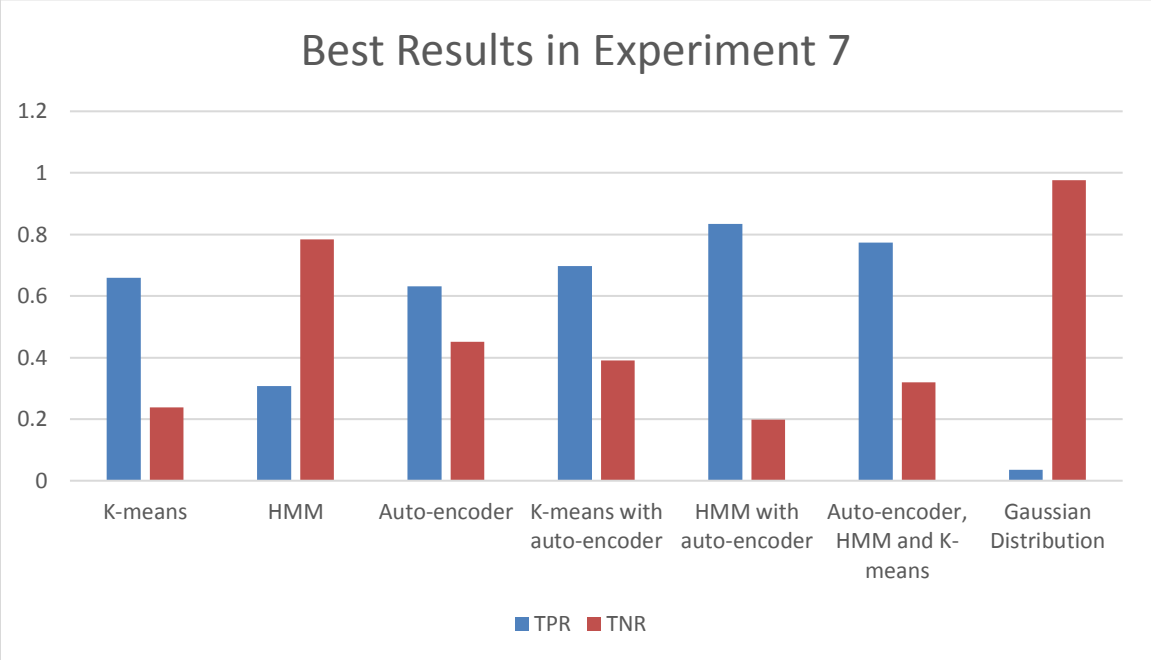

Figure 4.27: The Best Results in Experiment 7

4.4.8 Experiment 8 – Santander Customer Transaction Dataset

Experiment eight was implemented on a Santander Customer Transaction dataset. There are 200000 observations and each observation is described with 202 features. Table 4.48 describes some of the dataset characteristics which are provided by Santander Bank. An anonymized dataset containing numeric feature variables, the binary target column, and a string "ID_code" column is provided. The task is to predict the value of target column.

Table 4.48: Dataset 9 Description

Dataset name	Santander Customer Transaction dataset
Dataset features number	202
Dataset observation number	200K
Dataset Date	------
Dataset place	Spain
Normal - Anomalous percentage	89.95 - 10.05%

To visualize the dataset, the histogram function in Python was applied on the dataset, four features was shown in Figure 4.28 as sample:

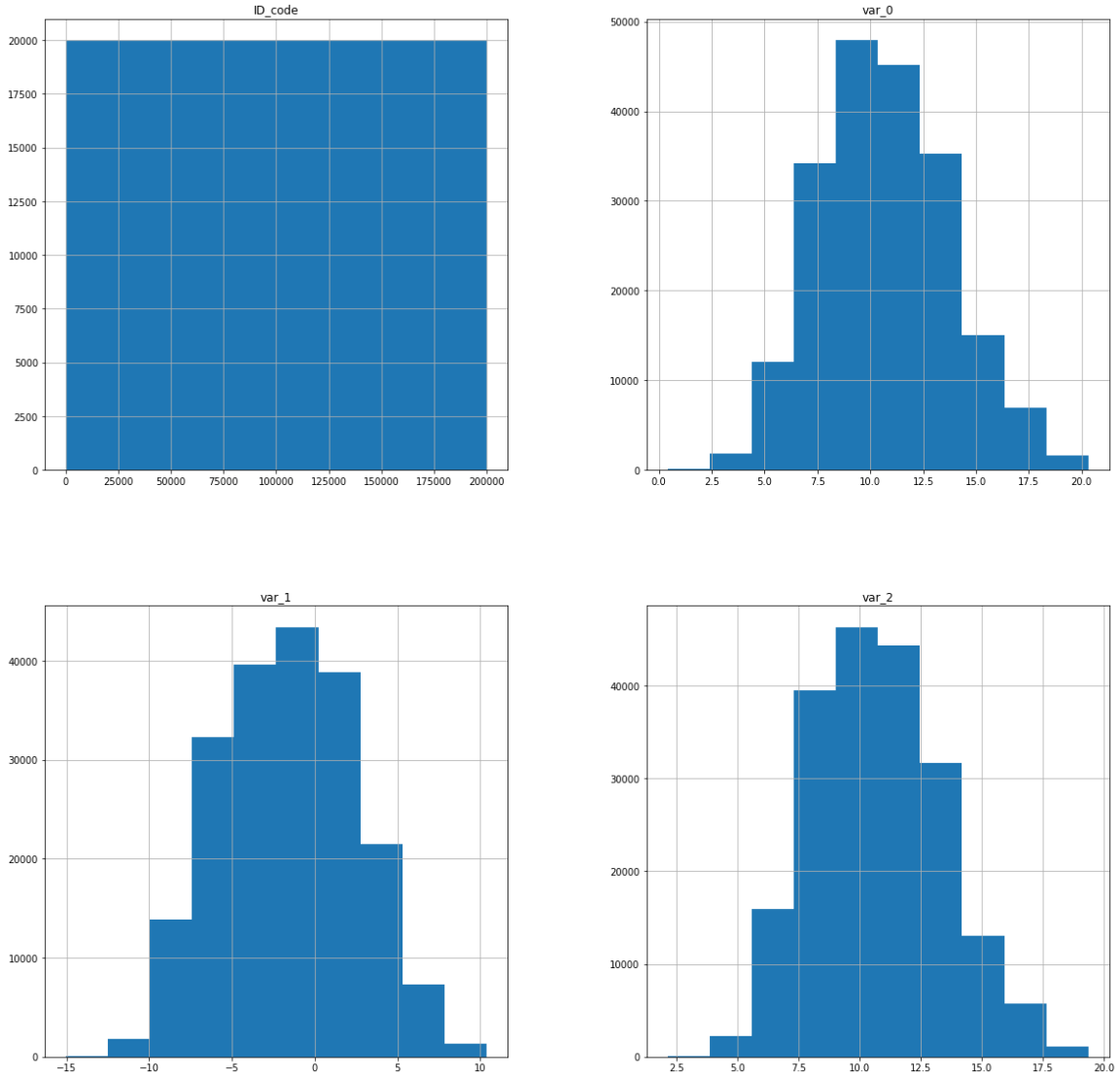

Figure 4.28: Features Histogram for Dataset 8

The Dataset is ready. Only one of the columns (ID_code) was described by letters. The letters are changed into suitable numbers. There are no NAN values and all other features are numbers. Finally, Feature importance was applied for applying PCA dimensional reduction. The features were sorted in term of importance to the target using extra tree classifiers, twenty features are shown in Figure 4.29 as sample. Additionally, a comparison was provided between data features in Appendix K.

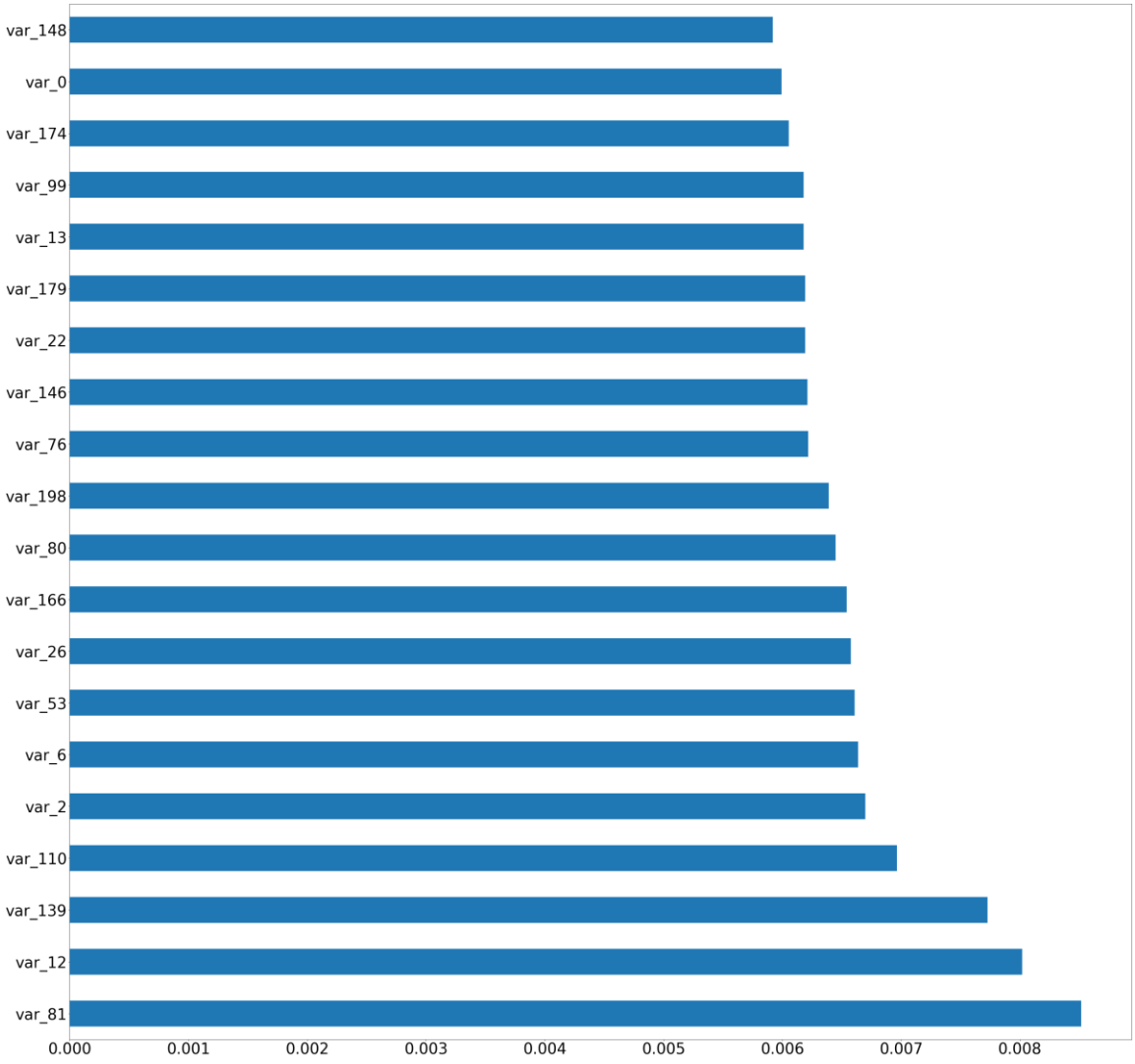

Figure 4.29: Feature Importance for Dataset 8

Default proposed models were applied between four assumptions for comparison between normalization and dimensional reduction. Table 4.49 shows the results with the four assumptions for Dataset 8. In the K-means model, the best result for TNR and TPR was achieved by applying the first and the third assumptions among the four assumptions. So, the first assumption was chosen as the best assumption to be applied for tuning parameters.

In the HMM model, Table 4.49 shows that the second assumption got the highest values in TNR and TPR among all assumptions of 51% and 54% respectively. So, the second assumption was chosen to be applied for tuning parameters.

In the Auto-Encoder model, all of the assumptions got the highest TPR except the second assumption which got the highest TNR. So, the fourth assumption was chosen to be applied for tuning parameters because it applies two preprocessing methods and there was no comparison in the results. In the Gaussian Distribution model, the fourth assumption was chosen to be applied for tuning parameters because it was applying two preprocessing methods and all the results are same. Finally, for more results such as F1 score and RMSE in this part refer to Appendix A, Appendix B, Appendix C, Appendix D.

Table 4.49: Results for Dataset 8 based on Four Assumptions

Models	Accuracy	TPR	TNR
Assumption 1: without normalization or dimensional reduction			
K-means	0.505159791	0.509702458	0.503891051
HMM	0.494709857	0.490596079	0.49585881
Auto-encoder	0.218318886	1	0
Gaussian	0.218318886	1	0
Assumption 2: with normalization only			
K-means	0.499923961	0.487610708	0.503362979
HMM	0.514545178	0.545924968	0.505780989
Auto-encoder	0.781681114	0	1
Gaussian	0.218318886	1	0
Assumption 3: with dimensional reduction only			
K-means	0.505203241	0.509702458	0.503946637
HMM	0.49460123	0.490596079	0.495719844
Auto-encoder	0.218318886	1	0
Gaussian	0.218318886	1	0
Assumption 4: with both normalization and dimensional reduction			
K-means	0.490951357	0.471589213	0.496359088
HMM	0.505377045	0.509005871	0.504363535
Auto-encoder	0.218318886	1	0
Gaussian	0.218318886	1	0

The best result for normal detection accuracy was 60% with low abnormal detection accuracy in 1 random state with one iteration. However, ten iterations in 1 random state gives the highest abnormal detection accuracy of 64% with acceptable normal accuracy

around 50%. Overall, the K-means model result has a moderate accuracy in this dataset. Table 4.50 summarized all K-means results. Finally, for more results such as F1 score and RMSE in this part refer to Appendix HH.

Table 4.50: K-means Results for Dataset 8

Tuning Parameters		**Evaluations**		
Max Iter	**Random State**	**Accuracy**	**TPR**	**TNR**
1	0	0.515196941	0.481938501	0.524485825
10	0	0.498229377	0.50771221	0.495580878
1	42	0.479632406	0.433674993	0.492468038
10	42	0.500814704	0.533883969	0.491578655
1	1	0.552955745	0.350781172	0.609421901
10	1	0.526993852	0.649616877	0.49274597
1	2	0.500271568	0.458354065	0.511978877
10	2	0.491516218	0.45815504	0.500833797
1	3	0.484303374	0.484923873	0.484130072
10	3	0.47661257	0.374763658	0.505058366
1	4	0.50087988	0.421235944	0.523123958
10	4	0.502922071	0.517165887	0.498943858
1	5	0.478111625	0.588018708	0.447415231
10	5	0.505007713	0.54144691	0.494830461
1	13	0.459927437	0.597571898	0.421484158
10	13	0.518412305	0.562444024	0.506114508
1	14	0.447913272	0.556473281	0.417593107
10	14	0.502031328	0.526321027	0.49524736
1	90	0.518325404	0.457259429	0.535380767
10	90	0.503704186	0.512488805	0.501250695
1	91	0.420843381	0.510299532	0.39585881
10	91	0.481761498	0.437456463	0.494135631
1	200	0.544548002	0.522041994	0.550833797
10	200	0.484064394	0.450890636	0.493329628
1	250	0.533620109	0.405512986	0.569399666
10	250	0.490538574	0.451189173	0.501528627

The results in this model showed better detections than K-means results in term of accuracy. It has higher accuracy for both normal and abnormal detection. The highest result for normal detection accuracy of 93% has a “diag” covariance type with very low abnormal accuracy. The highest result for both abnormal detection accuracy of 71% has a “full” covariance type with good abnormal accuracy of 50%. The other results were varied with “spherical” and “tied” covariance type and some of them gave a satisfactory accuracy level

for both, as shown in Table 4.51. Finally, for more results such as F1 score and RMSE in this part refer to Appendix II.

Table 4.51: HMM Results for Dataset 8

Tuning Parameters				**Evaluations**		
Covariance type	**N iter**	**algorithm**	**Tol**	**Accuracy**	**TPR**	**TNR**
Spherical	5k	viterbi	0.1	0.507679941	0.451786247	0.523290717
Diag	5k	viterbi	0.1	0.751895544	0.114339735	0.929961089
Tied	5k	viterbi	0.1	0.531708271	0.623743656	0.506003335
Full	5k	map	0.1	0.525103739	0.634291969	0.494608116
Spherical		viterbi		0.495491972	0.451587223	0.507754308
Diag		viterbi		0.499076669	0.484724848	0.503085047
Tied		viterbi		0.494840209	0.485819485	0.497359644
Full	5k	viterbi	0.1	0.475548024	0.382625137	0.501500834
Spherical	5k	map	0.1	0.489191597	0.482237039	0.491133963
Diag	5k	map	0.1	0.430902257	0.208080406	0.493135075
Tied	5k	map	0.1	0.51387169	0.518360036	0.512618121
Full	5k	map	0.1	0.492233157	0.493879988	0.491773207
Spherical		map		0.499141845	0.490297542	0.501612007
Diag		map		0.469986313	0.352273858	0.502862702
Tied		map		0.490821004	0.465717982	0.497832129
Full		map		0.550131439	0.712011145	0.5049194
Spherical	5k	viterbi		0.495491972	0.451587223	0.507754308
Spherical	5	viterbi	0.1	0.49588303	0.491989253	0.496970539

The auto-encoder results were tuned using the following parameters: number of epochs, batch size, input dimension, encoding dimension, hidden dimension for layer 1, hidden dimension for layer 2, activation function, learning rate, and threshold. The best results were obtained by varying the threshold values, as shown in Table 4.52. The highest normal and abnormal detection accuracies of 55 and 63% were obtained with one threshold value. Most of the other results have 0% abnormal accuracy and 100% normal detection. Finally, for more results such as F1 score and RMSE in this part refer to Appendix JJ.

Table 4.52: Auto-Encoder Model Results for Dataset 8

Tuning Parameters					**Evaluations**	
Encoding _dim	**Hidden _dim1**	**Hidden _dim2**	**activation**	**threshold**	**TPR**	**TNR**
18	10	6	tanh	4	0	1
18	10	6	tanh	4	0	1
32	16	8	tanh	4	0	1
10	5	2	tanh	4	0	1
5	2	1	tanh	4	0	1
5	3	1	tanh	4	0	1
50	20	10	tanh	4	0	1
5	2	1	sigmoid	4	0	1
5	2	1	hard_ sigmoid	4	0	1
5	2	1	exponential	4	0	1
5	2	1	linear	4	0	1
5	2	1	tanh	3	0	1
5	2	1	tanh	2	0	1
5	2	1	tanh	4	0	1
5	2	1	tanh	5	0	1
5	2	1	linear	4	9.95E-05	1
5	2	1	tanh	1	0.629714399	0.556031128
5	2	1	tanh	4	0	1
5	2	1	tanh	4	0	1

The results of the rest of the models were shown in Table 5.1. Auto-Encoder with K-means model gave more accuracy than the K-means and Auto-Encoder. Auto-Encoder with HMM model gave more accuracy level than auto-encoder and HMM. The combination model between the three model (K-means, HMM, and Auto-Encoder) gave approximately the same results compared with the previous two models. Comparing these results with HMM, Auto-Encoder, and K-means results did not have better results than the previous model. Finally, the Gaussian Distribution model gave an outstanding TPR which shows that the Gaussian distribution model has a high ability to classify the abnormal instances in this dataset.

Table 4.53: Results of Four Models for Dataset 8

Evaluations						
K-means with Auto-encoder Model Results						
Accuracy	**Precision**	**Recall**	**F1-score**	**RMSE**	**TPR**	**TNR**
0.578861612	0.586002557	0.625910 6	0.545982 268	0.64895176 1	0.7094238 23	0.54239 7376
0.588985444	0.587092314	0.627581 004	0.552634 617	0.64110416 9	0.6960891 63	0.55907 2844
0.581121008	0.586352127	0.626459 308	0.547534 249	0.64720861 5	0.7069360 14	0.54598 2602
0.581763671	0.585912907	0.625829 969	0.547721 944	0.64671193 7	0.7040501 54	0.54760 9783
0.589345847	0.586107776	0.626125 718	0.552358 554	0.64082302 8	0.6914120 81	0.56083 9355
HMM with Auto-encoder Model Results						
Accuracy	**Precision**	**Recall**	**F1-score**	**RMSE**	**TPR**	**TNR**
0.341472952	0.376349 681	0.321984 431	0.309248 355	0.81149 6795	0.287391 78	0.35657708 2
0.631674995	0.621152 625	0.677227 844	0.595067 235	0.60689 7854	0.758085 382	0.59637030 7
0.627634152	0.620444 28	0.676328 565	0.592247 983	0.61021 787	0.762762 464	0.58989466 7
0.339025397	0.373952 612	0.318554 371	0.306740 951	0.81300 3446	0.282217 136	0.35489160 6
0.630363467	0.623301 1	0.680512 291	0.595297 369	0.60797 7412	0.769529 306	0.59149527 5
K-means, HMM, and Auto-encoder Model Results						
Accuracy	**Precision**	**Recall**	**F1-score**	**RMSE**	**TPR**	**TNR**
0.513372005	0.517555 587	0.525712 175	0.473790 768	0.6976	0.5476	0.5038
Gaussian Distribution Model Results						
Accuracy	**Precision**	**Recall**	**F1-score**	**RMSE**	**TPR**	**TNR**
0.2183	0.1092	0.5	0.1792	0.884127318	1	0

In conclusion of experiment eight, the best results for each model was represented in Figure 4.30. The Gaussian distribution model got the highest TPR with unacceptable TNR. But the highest TNR and acceptable TPR was with HMM with Auto-encoder model. So, HMM with Auto-encoder model was considered as the best result in this experiment.

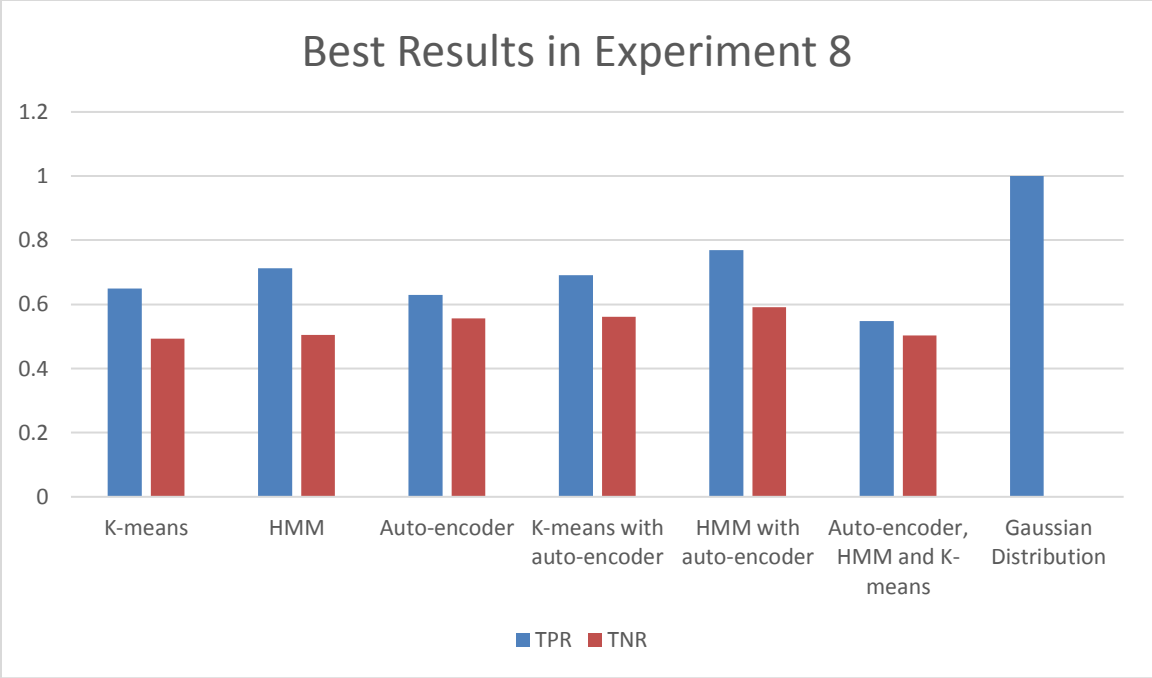

Figure 4.30: The Best Results in Experiment 8

4.4.9 Experiment 9 - Prudential Life Insurance Assessment Dataset

Experiment nine was implemented on a Prudential Life Insurance Assessment dataset. There are 59381 observations, each of which is described by 128 features. Table 4.55 describes some of the dataset characteristics which are provided by Prudential, one of the largest issuers of life insurance in the USA. In a one-click shopping world with everything on-demand, the old method of life insurance applications is antiquated. Customers provide extensive information to identify risk classification and eligibility, including scheduling medical exams, a process that takes an average of 30 days. The result is that people are turned off. That's why only 40% of U.S. households own individual life insurance. Prudential wants to make it quicker and less labor intensive for new and existing customers to get a quote while maintaining privacy boundaries. By developing a predictive model that accurately classifies risk using a more automated approach, you can greatly impact public perception of the industry. The results will help Prudential better understand the predictive power of the data points in the existing assessment, enabling us to significantly streamline the process. This dataset provided over a hundred variables describing attributes of life insurance applicants. The task is to predict the "Response"

variable for each ID in the test set. "Response" is an ordinal measure of risk that has 8 levels. Table 4.54 shows a features discretion in dataset 9.

Table 4.54: Data Features Description for Dataset 9

Variable	**Description**
ID	A unique identifier associated with an application.
Product_Info_1-7	A set of normalized variables relating to the product applied for
Ins_Age	Normalized age of applicant
Ht	Normalized height of applicant
Wt	Normalized weight of applicant
BMI	Normalized BMI of applicant
Employment_Info_1-6	A set of normalized variables relating to the employment history of the applicant.
InsuredInfo_1-6	A set of normalized variables providing information about the applicant.
Insurance_History_1-9	A set of normalized variables relating to the insurance history of the applicant.
Family_Hist_1-5	A set of normalized variables relating to the family history of the applicant.
Medical_History_1-41	A set of normalized variables relating to the medical history of the applicant.
Medical_Keyword_1-48	A set of dummy variables relating to the presence of/absence of a medical keyword being associated with the application.
Response	This is the target variable, an ordinal variable relating to the final decision associated with an application

The following variables are all categorical (nominal): Product_Info_1, Product_Info_2, Product_Info_3, Product_Info_5, Product_Info_6, Product_Info_7, Employment_Info_2, Employment_Info_3, Employment_Info_5, InsuredInfo_1,

InsuredInfo_2, InsuredInfo_3, InsuredInfo_4, InsuredInfo_5, InsuredInfo_6, InsuredInfo_7, Insurance_History_1, Insurance_History_2, Insurance_History_3, Insurance_History_4, Insurance_History_7, Insurance_History_8, Insurance_History_9, Family_Hist_1, Medical_History_2, Medical_History_3, Medical_History_4, Medical_History_5, Medical_History_6, Medical_History_7, Medical_History_8, Medical_History_9, Medical_History_11, Medical_History_12, Medical_History_13, Medical_History_14, Medical_History_16, Medical_History_17, Medical_History_18, Medical_History_19, Medical_History_20, Medical_History_21, Medical_History_22, Medical_History_23, Medical_History_25, Medical_History_26, Medical_History_27, Medical_History_28, Medical_History_29, Medical_History_30, Medical_History_31, Medical_History_33, Medical_History_34, Medical_History_35, Medical_History_36, Medical_History_37, Medical_History_38, Medical_History_39, Medical_History_40, Medical_History_41

The following variables are continuous: Product_Info_4, Ins_Age, Ht, Wt, BMI, Employment_Info_1, Employment_Info_4, Employment_Info_6, Insurance_History_5, Family_Hist_2, Family_Hist_3, Family_Hist_4, Family_Hist_5

The following variables are discrete: Medical_History_1, Medical_History_10, Medical_History_15, Medical_History_24, Medical_History_32 Medical_Keyword_1-48 are dummy variables.

Table 4.55: Dataset 9 Description

Dataset name	Prudential Life Insurance Assessment dataset
Dataset features number	128
Dataset observation number	59381
Dataset Date	------
Dataset place	USA
Normal - Anomalous percentage	74.4 - 25.6%

To visualize the dataset, the histogram function in Python was applied on the dataset, four features are shown in Figure 4.31 as sample:

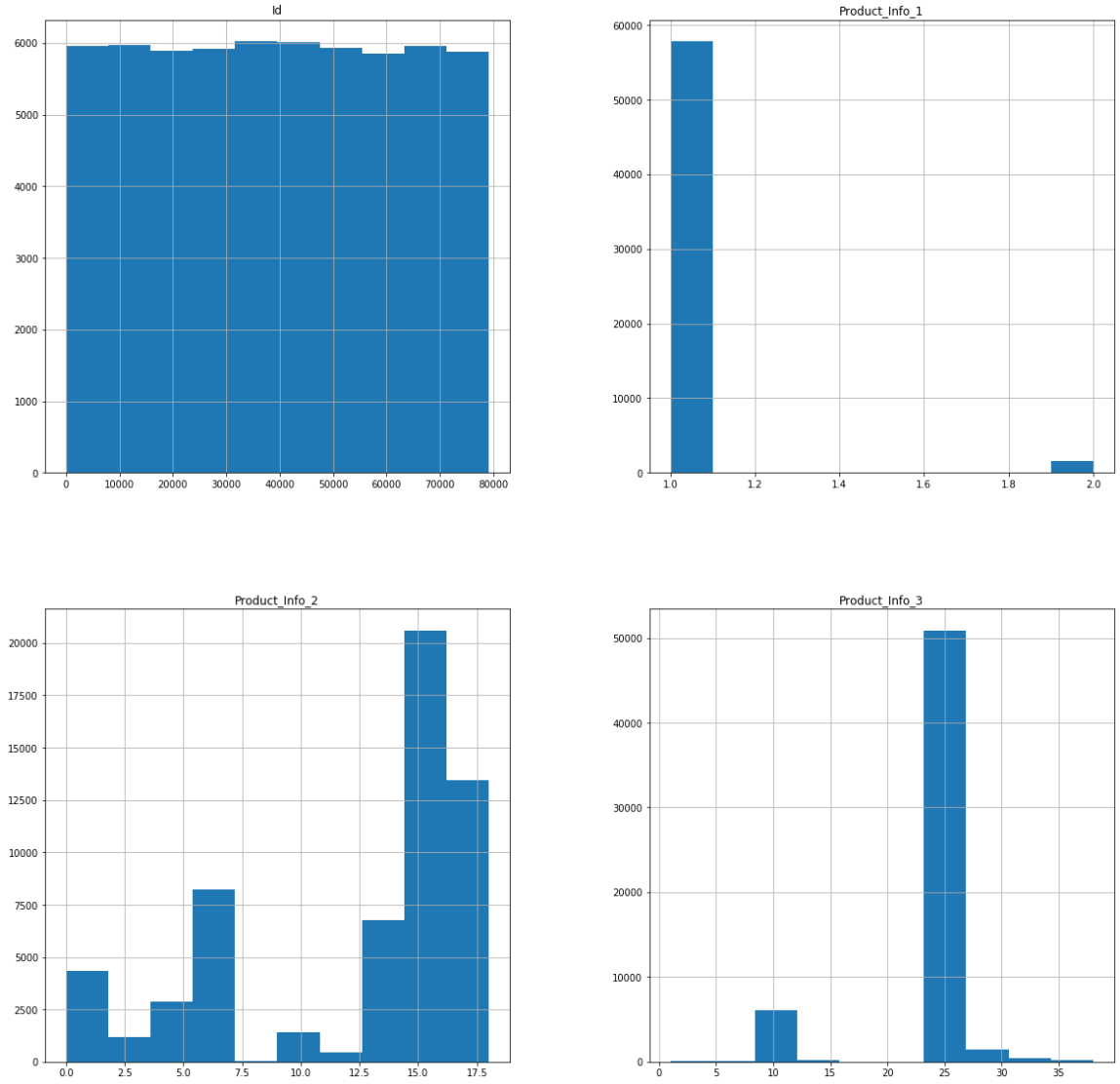

Figure 4.31: Features Histogram for Dataset 9

Some of the features have letter or word representations such as Product_Info_2. These features were replaced with a proper numeric feature. Other features have NAN values. These values were filled with the median values. “Response” was changed with two risk levels to present a binary classification. All other features are numbers and full with values. Finally, Feature importance was applied for applying PCA dimensional reduction. The features were sorted in term of importance to the target using extra tree

classifiers, twenty features are shown in Figure 4.32 as sample. Additionally, a comparison was provided between data features in Appendix L.

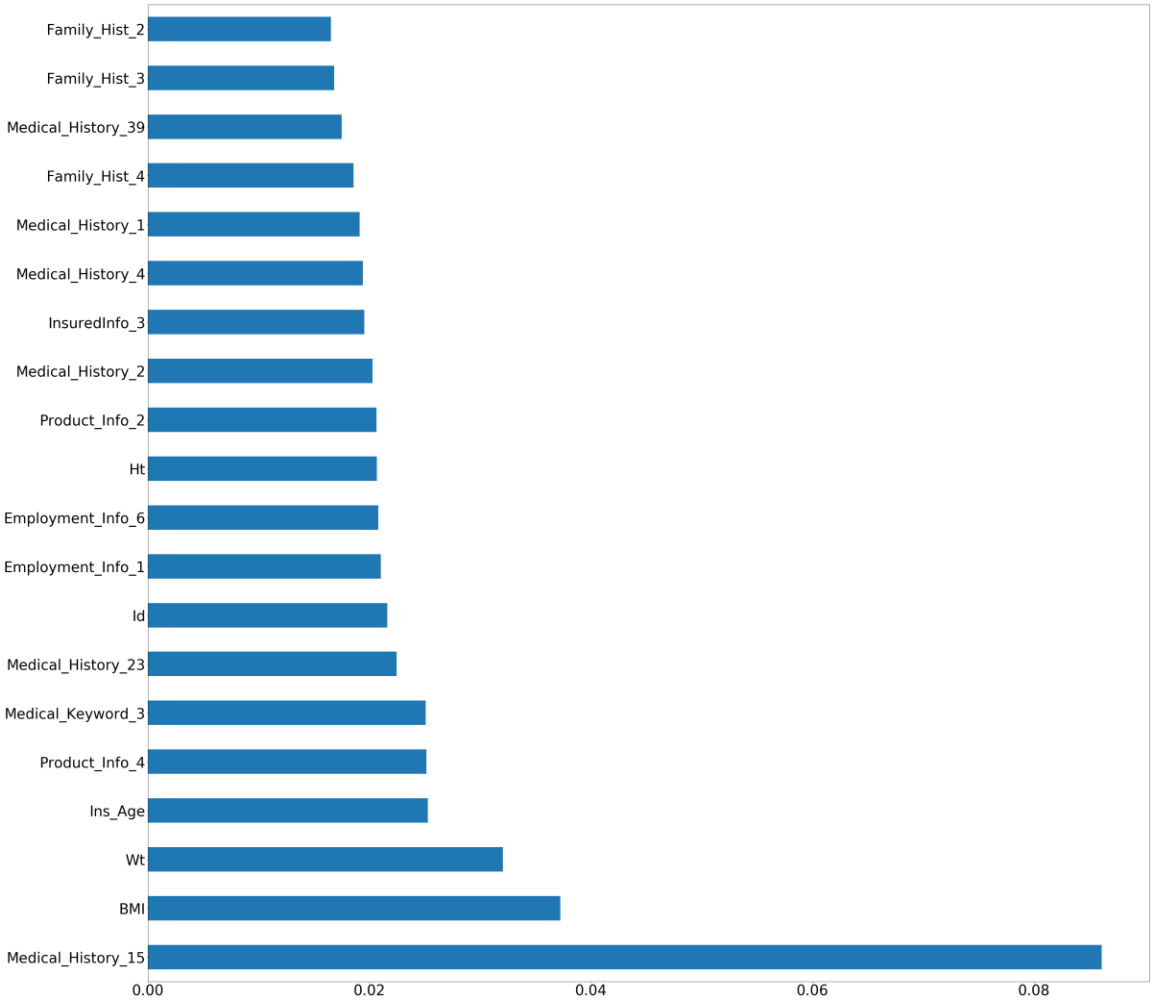

Figure 4.32: Feature Importance for Dataset 9

Default proposed models were applied between four assumptions for comparison between normalization and dimensional reduction. Table 4.56 shows the results with the four assumptions for Dataset 9. In the K-means model, the best result for TNR of 58% was by applying the second assumption among the four assumptions. However, the TPR for the second assumption was 43%. The highest TPR of 56% was with assumption four and a TNR of 41%. The first and third assumptions have balanced accuracies of 50% for both TNR and TPR. So, the third assumption was chosen as the best assumption to be applied for tuning parameters.

In the HMM model, Table 4.56 shows that the third assumption got the highest value in TPR of 71% among all assumptions of 54% TNR. Assumption four got the highest TNR of 58% and 64% TPR. So, the fourth assumption was chosen to be applied for tuning parameters.

In the Auto-Encoder model, the more suitable result for TPR and TNR was by applying the fourth assumption of 0.1% and 98% respectively. All other results have a 0% in either TNR or TPR. So, the fourth assumption was applied for tuning parameters because it has the highest TNR and TPR values. In the Gaussian Distribution model, the fourth assumption was chosen because it has the highest TPR and TNR results among all assumptions of 100% and 78% respectively. Finally, for more results such as F1 score and RMSE in this part refer to Appendix A, Appendix B, Appendix C, Appendix D.

Table 4.56: Results for Dataset 9 based on Four Assumptions

Models	Accuracy	TPR	TNR
Assumption 1: without normalization or dimensional reduction			
K-means	0.503711365	0.504078947	0.503395201
HMM	0.382453152	0.533157895	0.252829335
Auto-encoder	0.462399611	1	0
Gaussian	0.462399611	1	0
Assumption 2: with normalization only			
K-means	0.516853249	0.436052632	0.58635129
HMM	0.483572645	0.570921053	0.408442734
Auto-encoder	0.546665855	0.029078947	0.991851517
Gaussian	0.62204916	0.778947368	0.487098234
Assumption 3: with Dimensional Reduction only			
K-means	0.503711365	0.504078947	0.503395201
HMM	0.626733999	0.718026316	0.548211861
Auto-encoder	0.462338769	0.999868421	0
Gaussian	0.462399611	1	0
Assumption 4: with Both normalization and Dimensional Reduction			
K-means	0.483146751	0.563947368	0.41364871
HMM	0.607690436	0.640394737	0.579560887
Auto-encoder	0.556400584	0.061052632	0.982458126
Gaussian	0.886164517	1	0.788252603

The best result for TPR has outstanding accuracy around 97% in 14 random states with very low TNR. The highest TNR was 64% with an acceptable TPR of 53% in 90

random states. The other results have less accuracy. Table 4.57 summarized all K-means results. Finally, for more results such as F1 score and RMSE in this part refer to Appendix KK.

Table 4.57: K-means Results for Dataset 9

Tuning Parameters		**Evaluations**		
Max Iter	**Random State**	**Accuracy**	**TPR**	**TNR**
1	0	0.415551229	0.563157895	0.288592123
10	0	0.516853249	0.436052632	0.58635129
1	42	0.508335361	0.442368421	0.565074694
10	42	0.516853249	0.436052632	0.58635129
1	1	0.489596009	0.471447368	0.505205976
10	1	0.483146751	0.563947368	0.41364871
1	2	0.5183743	0.716052632	0.348347669
10	2	0.483146751	0.563947368	0.41364871
1	3	0.462703821	0.911578947	0.076618379
10	3	0.511742516	0.574605263	0.457673155
1	4	0.501825262	0.724736842	0.310095066
10	4	0.483146751	0.563947368	0.41364871
1	5	0.521720613	0.511842105	0.530217293
10	5	0.483146751	0.563947368	0.41364871
1	13	0.484424434	0.390657895	0.565074694
10	13	0.516853249	0.436052632	0.58635129
1	14	0.459905086	0.965526316	0.025011317
10	14	0.483876856	0.566315789	0.41296967
1	90	0.592966659	0.533289474	0.644296062
10	90	0.483146751	0.563947368	0.41364871
1	91	0.489717693	0.497105263	0.483363513
10	91	0.516853249	0.436052632	0.58635129
1	200	0.537722073	0.513947368	0.558171118
10	200	0.483146751	0.563947368	0.41364871
1	250	0.566439523	0.569473684	0.563829787
10	250	0.516853249	0.436052632	0.58635129

The results in this model showed better detections than K-means results in terms of accuracy. It has higher accuracy for both normal and abnormal detection. The highest result for both normal and abnormal detection of 74 and 99% respectively has a “full” covariance type. There were some results with less accuracy such as ‘spherical’ with ‘viterbi’ in covariance type and algorithm respectively. The other results were varied with “tied” and “diag” covariance type and some of them gave a satisfactory accuracy level for both, as

shown in Table 4.58. Finally, for more results such as F1 score and RMSE in this part refer to Appendix LL.

Table 4.58: HMM Results for Dataset 9

Tuning Parameters				**Evaluations**		
Covariance type	**N iter**	**algorithm**	**Tol**	**Accuracy**	**TPR**	**TNR**
Spherical	5k	viterbi	0.1	0.626673156	0.697894737	0.565414215
Diag	5k	viterbi	0.1	0.392309564	0.359605263	0.420439113
Tied	5k	viterbi	0.1	0.518191774	0.440131579	0.58533273
Full	5k	viterbi	0.1	0.859272329	0.999868421	0.738343142
Spherical		viterbi		0.626733999	0.698157895	0.565301041
Diag		viterbi		0.607690436	0.640394737	0.579560887
Tied		viterbi		0.481808226	0.559868421	0.41466727
Full	5k	viterbi	0.1	0.859272329	0.999868421	0.738343142
Spherical	5k	map	0.1	0.373326844	0.302105263	0.434585785
Diag	5k	map	0.1	0.392309564	0.359605263	0.420439113
Tied	5k	map	0.1	0.481808226	0.559868421	0.41466727
Full	5k	map	0.1	0.859272329	0.999868421	0.738343142
Spherical		map		0.373266001	0.301842105	0.434698959
Diag		map		0.607690436	0.640394737	0.579560887
Tied		map		0.481808226	0.559868421	0.41466727
Full		map		0.859272329	0.999868421	0.738343142
Spherical	5k	viterbi		0.626673156	0.697894737	0.565414215
Spherical	5	viterbi	0.1	0.373144317	0.298421053	0.43741512

The auto-encoder results wee tuned using the following parameters: number of epochs, batch size, input dimension, encoding dimension, hidden dimension for layer 1, hidden dimension for layer 2, activation function, learning rate, and threshold. The best results were obtained by varying the threshold values, as shown in Table 4.59. Most of the results gave the highest normal detection accuracy of 99 or 98% with close to 0% abnormal detection accuracy. Only two results have more abnormal accuracy of 30 and 60% with 87 and 65% normal detection. These results have two and one threshold values respectively. Finally, for more results such as F1 score and RMSE in this part refer to Appendix MM.

Table 4.59: Auto-Encoder Model Results for Dataset 9

Tuning Parameters					**Evaluations**	
Encoding _dim	**Hidden _dim1**	**Hidden _dim2**	**activation**	**threshold**	**TPR**	**TNR**
18	10	6	tanh	4	0.051315789	0.984495247
18	10	6	tanh	4	0.048684211	0.985740154
32	16	8	tanh	4	0.049078947	0.985513807
10	5	2	tanh	4	0.056052632	0.983703033
5	2	1	tanh	4	0.06	0.982231779
5	3	1	tanh	4	0.058684211	0.983023993
50	20	10	tanh	4	0.040789474	0.988909009
5	2	1	sigmoid	4	0.038947368	0.990606609
5	2	1	hard_ sigmoid	4	0.0575	0.982458126
5	2	1	exponential	4	0.058815789	0.982231779
5	2	1	linear	4	0.059868421	0.980986872
5	2	1	tanh	3	0.06	0.981213219
5	2	1	tanh	4	0.06	0.981213219
5	2	1	tanh	4	0.055526316	0.983023993
5	2	1	tanh	5	0.131315789	0.951901313
5	2	1	linear	2	0.297105263	0.869624264
5	2	1	tanh	1	0.597236842	0.654142146
5	2	1	tanh	4	0.025526316	0.991851517
5	2	1	tanh	4	0.056052632	0.982684473

The results of the rest of the models re shown in Table 4.60. Auto-Encoder with K-means model did not give more accuracy than the K-means or Auto-Encoder. Auto-

Encoder with HMM model did not give more accuracy level because the results of the two models separately were very high. The combination model between the three model (K-means, HMM, and Auto-Encoder) gave better results compared with auto-encoder and K-means models. Comparing these results with HMM results did not have better results. Finally, the Gaussian Distribution model gave an outstanding TNR and TPR of 79% and 100% respectively which shows that the Gaussian distribution model has high ability to classify the normal and abnormal instances in this dataset.

Table 4.60: Results of Four Models for Dataset 9

Evaluations						
K-means with Auto-encoder Model Results						
Accuracy	**Precision**	**Recall**	**F1-score**	**RMSE**	**TPR**	**TNR**
0.525582527	0.52117368	0.520955581	0.520723639	0.688779698	0.459473684	0.582437479
0.474417473	0.47882632	0.479044419	0.474077355	0.724970708	0.540526316	0.417562521
0.465413397	0.468836511	0.468966837	0.465337373	0.731154295	0.516184211	0.421749462
0.473168654	0.47735	0.477548426	0.472905853	0.725831486	0.535789474	0.419307379
0.481990752	0.487258974	0.487465214	0.481313289	0.719728593	0.560263158	0.41466727
HMM with Auto-encoder Model Results						
Accuracy	**Precision**	**Recall**	**F1-score**	**RMSE**	**TPR**	**TNR**
0.537628521	0.268814261	0.5	0.349647859	0.679979028	0	1
0.380787248	0.38218108	0.381726741	0.380623818	0.786900725	0.394210526	0.369242956
0.380787248	0.382401541	0.382039853	0.380696474	0.786900725	0.398684211	0.365395496
0.462399611	0.231199805	0.5	0.316192378	0.733212377	1	0
0.379654417	0.380832879	0.380304377	0.379407266	0.787620202	0.388947368	0.371661385
K-means, HMM, and Auto-encoder Model Results						
Accuracy	**Precision**	**Recall**	**F1-score**	**RMSE**	**TPR**	**TNR**
0.623387686	0.622735817	0.613202539	0.610550906	0.6137	0.4778	0.7486
Gaussian Distribution Model Results						
Accuracy	**Precision**	**Recall**	**F1-score**	**RMSE**	**TPR**	**TNR**
0.8862	0.9012	0.8941	0.886	0.337395144	1	0.788252603

In conclusion of experiment nine, the best results for each model was represented in Figure 4.33. The Gaussian distribution model got the highest TPR with acceptable TNR. But the highest TNR and unacceptable TPR was with HMM with Auto-encoder model. So, the Gaussian distribution model was considered as the best result in this experiment.

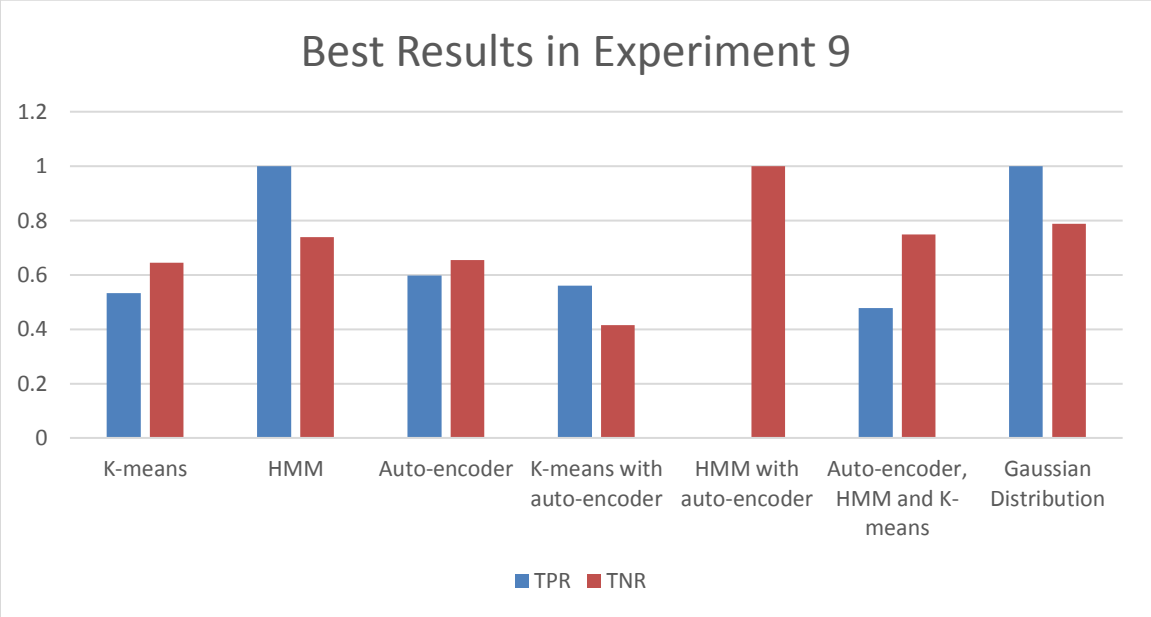

Figure 4.33: The Best Results in Experiment 9

4.4.10 Results Summary and Experiments Conclusion

The total number of instances overall the nine experiments was around 2 million exactly 1995669 observations. Table 4.61 summarizes the best model for the nine experiments. As it is shown in the experiment results, if a model is considered as the best model it does not mean the other models have bad results. In other words, most of the models, especially the combined models, detect the anomalies. However, some cases have poorly detection for specific experiments. The variety of best models gives an indication for a variety of applications, dimensions, and data types. Some models are only appropriate for some types of problems and can handle a limited data dimension.

Table 4.61: Best Model per Experiment

	K-means	HMM	Auto-encoder	K-means with auto-encoder	HMM with auto-encoder	Auto-encoder, HMM and K-means	Gaussian Distribution
Experiment 1						√	
Experiment 2		√					
Experiment 3					√		
Experiment 4			√				
Experiment 5	√						
Experiment 6			√				
Experiment 7				√			
Experiment 8					√		
Experiment 9							√

As shown in Figure 4.34, experiment 6 achieved the highest TPR and TNR. Most of the results in the experiments have good results. In total, six cases have the highest results which are three TPR and three TNR in experiments 2, 3, 4, 6, and 9. The lowest two cases of one TNR and one TPR were in experiment 3 and 7 respectively. The other results are achieved after the tuning process and the best model is chosen.

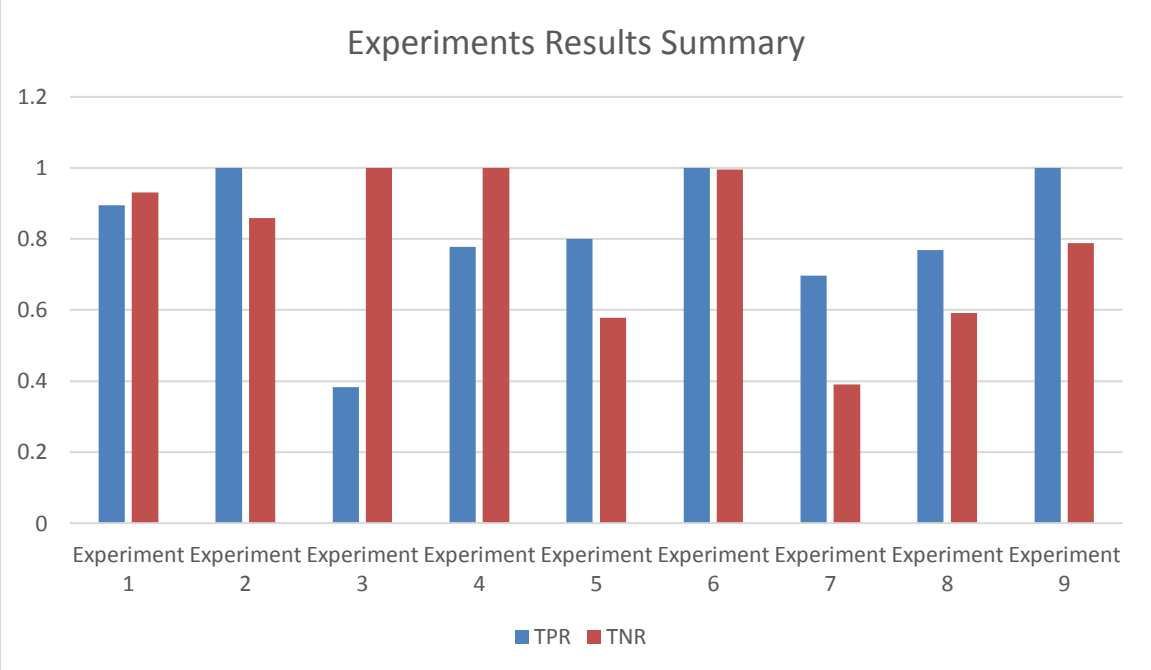

Figure 4.34: TNR and TPR for the highest result in every Experiment.

Chapter 5

5 User Authentication

This chapter will explain the new authentication method of “something you do” as a background for this chapter. It also proposes the user profile that will be generated using the results from the previous experiment in chapter 4. This chapter will also provide new mechanisms of producing the challenging questions based on the generated user profiles. Finally, it will explain a strong example for the authentication process through these questions.

5.1 “Something you do”-Based Authentication

In today’s world, security questions have become more popular in user authentication research fields. User authentication is a process of ensuring confidentiality of data that is claimed by a user for a system entity [87]. The challenge of the authentication process is to distinguish between legal or illegal authentication requests. In other words, the usage of a user authentication technique is to ensure that only the permitted user can access the data from the identification node [88]. Interestingly, various private and sensitive data is usually stored on the user’s account or system. Furthermore, if the account is unlocked, it is easy for attackers to steal the user's sensitive information, such as identity, photos and credit card information. Most user authentication methods are developed based on challenge and response questions to protect the user against any attack [89]. User authentication has a variety of methods that can identify the valid users in protected resources which can be classified broadly into four groups based on something the user “is”, “knows”, “has”, and “does”. “Something the user does” is one of the new user authentication process’s that has been researched in recent years. This employs the user’s activities such as Knowledge-based authentication (KBA) [34].

KBA is an authentication system in which the user should answer a set of security questions (or at least one) to be authorized. Generally, the security questions have two major categories; static and dynamic [35]. The static questions are the most commonly used, but it is considered a weak authentication method for three reasons [90]:

A. Security questions’ context does not apply for the user currently.

B. Users usually forget the answer content or formatting when they are selected at setup.

C. The correct answers are very guessable because they are common knowledge or researchable because they are found online or by asking.

One common application for static security questions is “Fallback Authentication” that is a backup for authentication techniques in the lost cases. Moreover, fallback authentication is usually used when people lose their authentication access due to changes or forgetting the authentication requirements such as forgetting a password or username [90]. Fallback authentication identifies the user through personal information and allows the authenticated user to re-access their resources [36]. However, static questions are a vulnerable way to ask in Fallback Authentication because the answers to these questions can be easily reachable with a quick Google search. Also, as more personal information is available in public records, it is becoming easier for attackers to retrieve this information through observational attacks, from social network apps, such as Facebook, Twitter, Instagram or even more professional websites like LinkedIn [37].

The second type of challenging questions more invulnerability than the first type due to the dynamic way of asking the questions. These Dynamic security questions are taking the lead in question generation based on user behaviour other ideas [35]. There are different ways to create these dynamic questions such as user Internet activities, a story creator, and autobiographical authentication [92].

The stronger way to produce a secure dynamic question achieves a more secure system against any fraudulent or abnormal activities through dynamic information. With the existence of dynamic information, the system may ask for a different set of questions to provide unique security questions [93]. Figure 5.1 summarizes the security questions types and some examples.

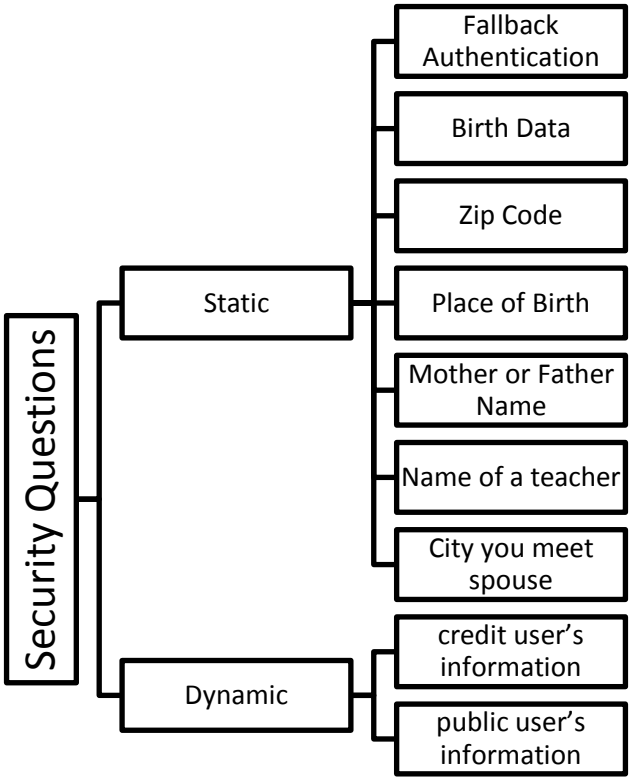

Figure 5.1: Security Questions Types and Examples

As a result, unique dynamic security questions should be investigated with several features:

A. A set of challenging questions without using highly guessable answers
B. Abnormal user activities
C. Using short term history or up-to-date
D. unrepeated questions

This new way of asking the dynamic security questions can be generated based on studying the abnormal activities of the user behaviour utilizing anomaly detection.

5.2 User Profile

The primary user profile's purpose is to use it as a Database for generating dynamic security questions. The proposed user profiles will be created based on anomaly detection results in chapter 4. When the data has been flagged as an anomaly, the data information

will be collected from the features such as location, time, amount, and so on and then will be used for user profile generation [94]. The user profile specification contains several features as shown in Table 5.1; the prime user identification, action description (credit card transaction for example), timestamp, expected user behaviour, briefly explains the anomalous user behaviour. Table 5.1 also shows the data type corresponding to the feature name.

Table 5.1: User Profile Specification Features

Feature name	User Identification	Time	Action	Observation	Expected Behaviour
Data Type	Numbers and characters	Numbers	String	String	String

The user identification could be the account number, user ID number, or any unique number that can identify the user from the data. The action description is a general feature type such as a credit card transaction, cash payment, or online purchase. The timestamp is a significant feature because it specifies the action time. It could be in many formats like minutes, seconds, or days depending on the data description. An example of the expected user behaviour could be any normal or regular activities regarding the user history such as a car with gas on a weekly basis, daily supermarket purchases with a small amount range, or a morning coffee purchase. Lastly, the anomalous user behaviour should be something that deviates from the expected behaviour such as gas filling on a daily basis, daily supermarket purchase with a huge amount, or an evening coffee purchase. These features are collected, presented, and analyzed to help the next user authentication step which will utilize these profiles efficiently to create the dynamic security questions.

Moreover, it only contains a feature that describes abnormal user behaviour and what is the expected user behaviour. In the next few tables, user profile samples are provided with the related experiment from chapter 4. For example, Table 5.2 shows a sample of the user profile detail from experiment 1. The user identification is 439, the timestamp is 6986 seconds, the anomalous user behaviour was a very early morning time around 1:56:43 am, and the expected time based on the user history is during the day time from 8 am to 9 pm.

Table 5.2: User Profiles Sample from Experiment 1

User Profile for Sample of Anomalous Data			
User Identification	Time Stamp (sec)	Observation	Expected Behaviour
ID-231	406	0 amount	More than 0
ID-439	6986	not expected time	during the day time
ID-349	9064	huge amount	normal range
ID-204	53937	far store branch from user home	the usual store is the nearest for this user
ID-007	56887	a new cvv code for the same usual card	the usual cvv code number
ID-127	57007	low amount	real amount
ID-114	62330	first time purchase from this category	no purchase from this category
ID-534	62467	many items from the same product	one is the usual of this product
ID-108	76867	different membership level from last time	last time was the first level
ID-093	84204	different home address	the old home address

Table 5.3 shows a sample of the user profile detail from experiment 2. For example, the user identification is 'C1350963410', the timestamp is 61 steps, the anomalous user behaviour was buying a children's toy for first time, and the expected purchase based on the user history is buying adult things.

Table 5.3: User Profiles Sample from Experiment 2

User Profile for Sample of Anomalous Data				
user Identification	time stamp (steps)	action	observation	expected behaviour
'C204205576'	0	bank payment	not expected time	during the day time
'C1273692645'	1	bank payment	huge amount for this product	normal range for this product is lower
'C225675370'	153	bank payment	first time purchases from this category	no purchases from this category
'C2044438336'	87	bank payment	unexpected a male purchase	usually a female purchase
'C1350963410'	61	bank payment	different age for this product	older customers buy this product

Table 5.4 shows a sample of the user profile detail from experiment 3. For example, the user identification is 2, the timestamp is 36 months, the anomalous user behaviour was working in the retirement age, and the expected status based on the user's history is retirement by this age.

Table 5.4: User Profiles Sample from Experiment 3

User Profile for Sample of Anomalous Data				
user Id	time (months)	action	observation	expected behaviour
ID- 4	12	credit history	unexpected increasing in saving account	during this time there is no increasing in saving money
ID-28	1	credit history	huge amount in checking account increased by one month	normal range for checking account in one month is small amount
ID- 32	2	credit history	young user age for the employment status	this age is usually unskilled employment

Table 5.5 shows a sample of the user profile detail from experiment 6. For instance, the user identification is 55, the timestamp or record time is 0.3 seconds, the anomalous user behaviour was the record time (0.3) is very low regarding the number of bytes which is 54540, and the expected bytes range based on the record history for low record time was from 6 to 410 bytes.

Table 5.5: User Profiles Sample from Experiment 6

User Profile for Sample of Anomalous Data				
user Identification	time (seconds)	action	observation	expected behaviour
ID- 55	0.3	Record	The record time is very low regarding the number of bytes	The normal record time much more for this number of bytes
ID- 1389	0.17	Record	the number of data bytes transferred from the destination to the source is very high	The normal number of bytes are much lower
ID - 65927	9	Record	the number of data bytes transferred from the source to the destination is very high	The normal range in the number of bytes does not include high numbers.

Table 5.6 shows a sample of the user profile detail from experiment 7. For example, the user identification is 563, the timestamp is 9 days, the anomalous user behaviour was an unexpected speed with 104 km/h, and the expected speed average based on the user history is 62.5 km/h.

Table 5.6: User Profiles Sample from Experiment 7

User Profile for Sample of Anomalous Data				
user Identification	time stamp (days)	action	observation	expected behaviour
ID - 84	9	Car driving	The driving region of this user is different	The normal driving region for this user in the user city
ID - 563	9	Car driving	The speed of this user is very high out of the normal range	The normal speed range for this user is low
ID - 2204	4	Car driving	An accident report for this user with the car	This user has a free accident history

Table 5.7 shows a sample of the user profile detail from experiment 9. For example, the user identification is 46185, there is no timestamp provided in this dataset, the anomalous user behaviour was a heart attack with an operation in recent medical history, and the expected health based on the user history is that the user has a free operation history and good health.

Table 5.7: User Profiles Sample from Experiment 9

User Profile for Sample of Anomalous Data				
user Identification	time stamp	action	observation	expected behaviour
ID - 49	NAN	Health record	The height is increased for this user	The age of this user has no height expected increasing
ID – 1023	NAN	Health record	The weight of this user decreased sharply.	The normal weight of this user much less than the last observed one
ID - 46185	NAN	Health record	Operation happened with this user last month	This user has good health without any operation

5.3 Creating an Individual User Profiles

Nowadays, the available user information is increasing rapidly which make it difficult for systems to quickly and automatically detect the abnormal users' actions. Users have a wide range of behaviours especially with different action types, and these users have a range of interests and patterns [106]. Building a user profile based on the system requirements is a solution that organizes massive user information and extracts the most important features. The definition of user profile stated as a description the user behaviors usually using user information such as user ID, time, action type, behaviour description, and so on [107] and [108]. The user profile approaches are employed with a specific structure that relates to system objectives to provide readable personalized results for each user. For example, if the system requires anomalous user information, then the user profile is built based on the anomalous users' actions. Also, one of the important user profile features is a dynamic updating feature which considers the changes of the users' actions over time [107].

In this thesis, user profiles are built based on the proposed anomaly detection system that provide the required results for the anomalous actions. We used a machine learning technique to detect the anomalous user's actions and then build a user database that will be fed automatically from the user and the machine learning.

A comparative study of user profiles will be presented before the user behavior modeling is explained. We will compare between the user profiles based on selected factors which will allow researchers to drive critical thinking ideas such as choosing a suitable profile structure for certain problems and conditions. The criteria of choosing the research papers depends on two shared factors: the user profile approach and anomaly detection problem. The result of this study is expressed in Table 5.8. Where BIDS is **B**ehavior **I**ntrusion **D**etection **S**ystem, DBMS is database management system, MSSQL is Microsoft SQL Server, UEBA is User and Entity Behavior Analytics, and VoIP is Voice over IP communication.

Table 5.8: User Profiles Comparison Table

Problem Field	Features Number	Profile Type	AD Technique	Profiler Tool
Network [107]	5	normal activity profile	K-means clustering	Behavior Detector
Securing Databases [108]	6	Role Administered Relational	Support vector machine	DBMS
Network [109]	3	Time-Variant Normal	Needleman-Wunsch	AD technique
cellular mobile networks [110]	4	normal profile	Rough Set	Rough Set
Hadoop File System [111]	5	Behavior-based profiles	K-means clustering	Eagle
Network [112]	4	User behavior profile	Apriori-k	MSSQL
User log [113]	5	User activities	BIDS detector	BIDS
Network [114]	8	Behavioral approach	K-means clustering	-------
Insider Threat [115]	3	user behavior	Neural networks	UEBA
Voice over IP communication [116]	3	Deep Packet Inspection	Support vector machine	VoIP

The user behaviour modeling is represented in three main parts as shown in Figure 5.2. Every part will be explained in the next subsections. The anomaly detection part produces the binary results for user behaviour. A database for every user is created based on the AD results. The total database for every user builds a user profile. Finally, a questionnaire will provide dynamic security questions based on the user profiles for user authentication purposes.

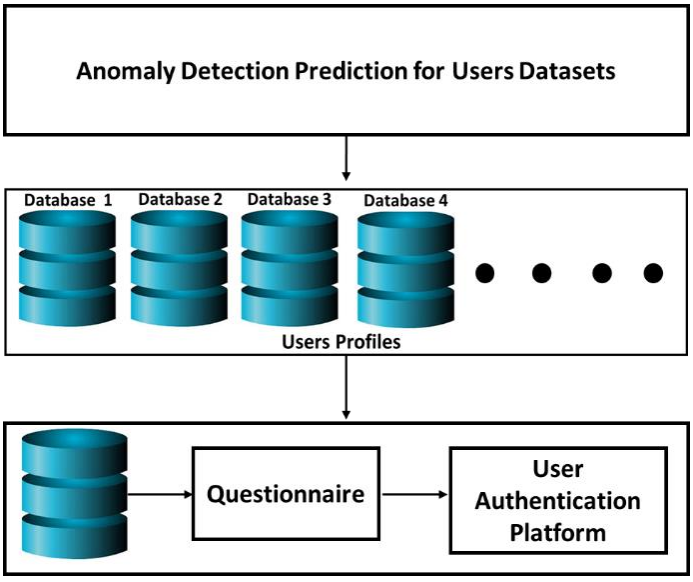

Figure 5.2:User Behaviour Modeling Diagram.

The anomaly detection model contains important steps to proceed for the user profile generation as shown in Figure 5.3. Firstly, the model is collecting Big Data based on user information that can represent a user's activity with unique identification. The Big Data is analyzed based on users before feature selection is applied. Feature selection is applied based on the user analysis to choose the most important feature that is related to the anomalous action not related to the user's personal information such as user ID. The preprocessing step contains any data preparation such as normalization and data splitting. Finally, the processing of anomaly detection technique to predict the binary results will be the input data to build the profiles.

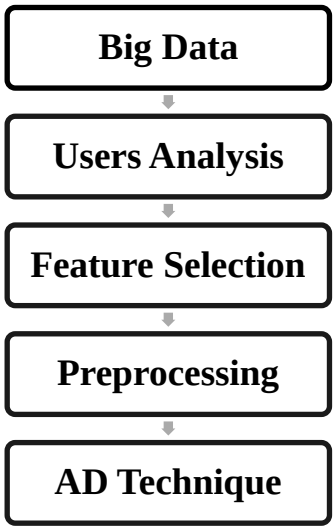

Figure 5.3: Anomaly Detection Model.

The database for every user is generated based on the anomaly detection results that are contained in a classification of normal and abnormal actions. In this research, the database is created only for the abnormal actions using the user profile structure that is proposed per user. The normal actions are also taken in consideration to calculate the normal or average values for any action type or features as shown in Figure 5.4.

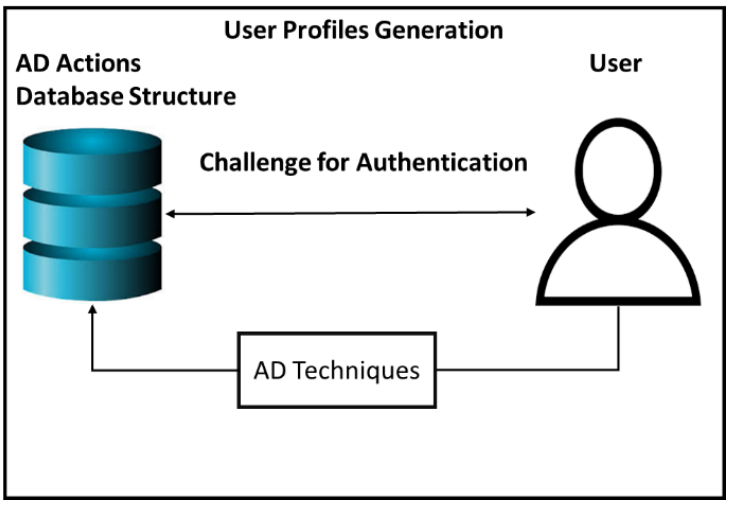

Figure 5.4: User Behavuior Modeling Diagram.

This process of creating the database is done automatically using the algorithm that is shown in the following description. Initially, the input data used the binary predictions

from anomaly detection techniques. Then the algorithm calculates the normal user pattern based on the normal actions. After that, the algorithm takes the abnormal instances with the related features. One of the features has the most effect that flagged this instance as an anomaly. This feature is determined and compared with the normal value. Finally, the user profile is built using the user ID, Action type, Time, unexpected observation, and the expected behavior. The user ID, Action type, and Time is written to the database from the original data. The unexpected observation is written using the most effect feature that is calculated in the algorithm. The expected behavior is written to the database using the normal values per feature that are computed previously. All these collected features in the database described the abnormal classified instances per user. The user profile is readable and ready for security question generation.

Algorithm 4: User Profile Creation
INPUT: binary predictions from anomaly detection technique
OUTPUT: User profile for abnormal observations

```
1  Begin
2      Read the input data from the model output (predictions)
3      Calculate the normal user pattern for every attribute from the normal instances
4      Separate the abnormal instances with the related attributes
5      Calculate the attribute that cause the abnormal instances
5      Build the user profile structure
6      Write the user information into the user profile structure form the original data
7      Write the abnormal observation into the user profile for every abnormal instance
8      Write the related expected user behavior to the abnormal observation
9      Repeat these steps for all users
10 End
```

The selected features for training the anomaly detection are the time step, Merchant, Category, Amount. The Zip Code and Zip merchant are not selected because they are the same for all users. The user representation features; Customer ID, Age, Gender, are not selected but it will be used in the user profile generation such as Customer ID. All the features are normalized and prepared through the preprocessing step to be ready for the AD technique. The final AD results contain a prediction of anomalous data per user. The algorithm detects the anomalies for every user. The total number of users are 4112 users. For example, the user with ID ‘C1093826151’ has 18 anomalous instances out of 167 instances. As a result, the normal instances for this user is 159 instances. Every abnormal instance will be described in the user profile with several features. The 159 normal instances will be studied to provide the related normal pattern for the user.

The user profiles are generated based on the user analysis using the anomaly detection techniques. The total number of user profiles that are generated in this dataset is 4112 which are the number of users. As a sample of a database that creates the user profiles, the anomalous user profile is presented in 0for the user with ID ‘C1093826151’. The database is readable and ready for generating the dynamic security question with the features that are specified in section D. The user profile for this user contains 18 rows which are the number of anomalous instances for this user with 5 columns that describe the abnormal action and the user information.

Table 5.9: a Sample of User Profile

User ID	Action	Time	Unexpected Observation	Expected Behavior
'C1093826151'	Transportation	2018-06-18	Time	2018-03-27
'C1093826151'	Bars and Restaurants	2018-06-24	Category	Transportation
'C1093826151'	Transportation	2018-05-27	Time	2018-03-27
'C1093826151'	Transportation	2018-06-11	Time	2018-03-27
'C1093826151'	Transportation	2018-03-29	Time	2018-03-27
'C1093826151'	Transportation	2018-06-03	Time	2018-03-27
'C1093826151'	Transportation	2018-05-29	Time	2018-03-27
'C1093826151'	Transportation	2018-05-15	Time	2018-03-27
'C1093826151'	Transportation	2018-06-13	Time	2018-03-27
'C1093826151'	Transportation	2018-06-02	Time	2018-03-27
'C1093826151'	Transportation	2018-06-28	Time	2018-03-27
'C1093826151'	Sports and Toys	2018-06-19	Merchant	'M348934600'
'C1093826151'	Transportation	2018-05-20	Amount	28.8007
'C1093826151'	Transportation	2018-04-29	Amount	28.8007
'C1093826151'	Transportation	2018-04-22	Time	2018-03-27
'C1093826151'	Transportation	2018-03-30	Time	2018-03-27
'C1093826151'	Transportation	2018-04-23	Time	2018-03-27
'C1093826151'	Transportation	2018-05-26	Time	2018-03-27

5.4 Challenging Questions

The proposed user authentication system is based on a "knowledge-based authentication" technique that uses a uniquely dynamic way to ask security questions. These security questions should have essential features to achieve a final robust authentication system. These features contain a set of challenging questions using short term personal history that are based on anomalous cases and not repeated. These questions are based on the anomalous data to allow only the user who can provide the answers for them. Short-term history is employed because it is imperative to keep the answers easy to remember only for the user and difficult to know for anyone else. However, if it is a long-term user history, it will be complicated for the user to remember the answers, particularly for dynamic and not static questions. Unrepeated questions are critical nowadays because hackers can find out answers. In other words, if hackers discover an answer, it will be dangerous to repeat the question.

The scenario of user authentication starts from the user profile information. The questions will be asked as a set of dynamic questions based on the information provided in the user profile database. It is supposed that only the user knows the answers to these questions because it is an abnormal observation and recent user history.

For example from experiment 1, if the time stamp was at a not expected time such as in '6986' sec which is around 1:56:43 am in the morning, we should ask the user about the time first "What was the time of your credit card transaction?" and then follow it by a set of questions about the location, amount and so on. The benefit of asking a set of questions is to add more security about the abnormal cases that nobody else would be expected to know. In other words, the user is the only person who knows all the information about the abnormal observation. Another sample is the money amount that user "349" showed was $1809.68 which is over this normal user range (100 - 500). The appropriate question will be "what was the amount of money in your recent transaction?" Lastly, the "007" user used a new CVV code '256' which is not the same usual CVV code '181' in the system. The following security question will be "What was your CVV code number for last credit card transaction?" The novelty of this approach is that instead of asking questions about the normal activities of the user (that can be figured out easily), we ask

questions about the recent abnormal actions of the user (that is hard to be guessed by others). It is worth mentioning that, each question is asked only once, and the questions set should be randomly chosen from a pool of candidate questions.

An example from experiment 2, if the product was bought from the user only once which is an unexpected product category from the history such as in 'es_otherservices' by the user 'C225675370', we should ask the user about the category first. “What was the type of the product in your last transaction?” and then follow it by a set of questions about the location, amount and so on. The benefit of asking a set of questions is to ensure that the user is answering and not someone else. If all the information for the abnormal case was provided correctly that means the user is correctly authenticated because the only person who knows all the information about the abnormal observation is the user. Another sample is the type of product 'es_sportsandtoys' was bought by user 'C2044438336' which is for his age range (50 – 60 years). It also shows based on his history that was once during 180-timestamps. The appropriate question will be “what did you buy in your last transaction?”

An example from experiment 3, the user with ‘ID - 28’ has a sharp increase in his checking account over 200 DM (Deutsche Mark; Germany currency) for one month and the normal checking range based on this user history is under 200 DM for one month. We should ask the user about the amount of money in the checking account first, “How much money do you have in your checking account?” and then follow it by a set of questions about the time of that increasing, account number and so on. Another sample is the employment status changed for user ‘ID - 32’ in one month which is not normal for this young age range (20 - 30) to have skilled employment based on the history. The appropriate question will be “What is you employment status now?”

An example from experiment 6, the user ‘ID - 55’ has a very short time recording of 0.3 which is not normal for this number of bytes ‘54540’. We should ask the user about the category first “How long was your last recording time?” and then follow it by a set of questions about the ID, destination name and so on. Another sample is that the user with ‘ID - 65927’ has an out of normal range in the number of bytes of ‘54540’ and the normal

bytes range is 6 - 410 bytes. The appropriate question will be “How long was your last recording time?”

An example from experiment 7, User with ID ‘563’ was driving with a speed of 104 km/h 9 days prior which is an abnormal speed range for this user (the normal range based on the user history is 40 – 85 and the average is 62.5). We should ask the user about the speed first “What was your speed while driving 9 ago?” and then follow it by a set of questions about the location, time and so on. Another sample is an accident is reported 4 days prior for user ‘ID - 2204’ which has a clean history of accidents. The appropriate question will be “where and when did your accident happened?”

An example from experiment 9, the medical record for user with ‘ID - 1023’ highly decreased in weight to 35 kg but before that it was 55 kg. We should ask the user about the weight first “What was your last weight?” and then follow it by a set of questions about the time, reason and so on. Another sample is for a heart operation for user ‘ID - 46185’ who, based on the medical records had no operation before. The appropriate question will be “what type and where did your operation occur?”

Chapter 6

6 Conclusion and Future Works

6.1 Conclusion

The research in user profile for Big Data-based applications has been increasing especially those utilizing anomaly detection techniques such as outlier detection, fault detection, computer system monitoring, and event detection in IoT devices. User trait modeling application lacks a robust implementation for anomaly detection. User trait models represent the user behaviour so that user variations in the system are noticed and interpreted. The reason of adoption in user trait modeling increases out of needing a continuous flow of high-volume data, that is not always available, to achieve high-accuracy detection. An existing user authentication framework provides an ambition for user trait modeling.

The main goal of this research is to present a solution model that designs and implements an anomaly detection technique suite for the user authentication framework. The solution model is designed from an investigation on Big Data for anomaly detection techniques. The investigation recommends three new classifications which are accomplished by combining three chosen Big Data V's with three anomaly detection factors that are related to the V's as follows:

1) Velocity with computational complexity classification includes the two types of algorithm time complexity; linear and quadratic and two types of data labels (supervised and unsupervised) for each time complexity type.

2) Variety with the natural types of data classification focuses on the data types such as time series, text, and media with providing a Big Data types and sources.

3) Volume with data features classification considers two major feature types which are univariate and multivariate.

Every classification defines the common machine learning (ML) techniques that are used in recent research. These classifications drew the outlines to choose the best model

fit with the best problem. The last part of this investigation was two comparison studies related to the data labels; supervised and unsupervised techniques, over a number of recent research papers which are compared after choosing the common ML models with defined comparison factors and several research paper conditions.

The main part of the solution model is provided with an anomaly detection model that contains a combination of several techniques that are suitable for the existing user authentication framework. The anomaly detection models are combined with several machine learning techniques; K-means, HMM, Auto-Encoder NN, and Gaussian distribution. In total, the applied models and techniques are seven; the four basic techniques without any combinations and three combined are as follows:

1) K-means is combined with Auto-encoder neural network which use the auto encoder for learning user behaviour and use K-means to differentiate between the normal and abnormal instances.

2) HMM is combined with Auto-encoder neural network that utilize auto-encoder to reproduce the data to learn the user pattern and utilize the HMM for detection purposes.

3) K-means is combined with HMM and Auto-encoder neural network to use the same purposes for HMM and auto-encoder. However, the K-means in this case is used to calculate the data probability parameters for HMM detection process.

Nine different experiments are applied to the proposed models and give a good detection result for each experiment. The applied experiments have a variety of fields such as financial payment systems, insurance systems (health, auto, and home), computer servers monitoring systems, and network transmission systems. The evaluation methods are chosen by applying most of them in this thesis such as confusion matrices, true positive rates (TPR), and true negative rates (TNR). Also, two algorithms are developed to ensure that the chosen evaluation methods match the needs of the user authentication framework.

From the results of the desired anomaly detection models, user profiles are generated as part of the solution model for the suitable experiments. The features of the user profiles were the same for all users in all used experiments in this part. A total of six

user profiles per experiment are designed and applied as databases for challenging questions. The final part of the solution model is providing a scenario of generating challenging questions based on the proposed user profiles. This scenario provides strong examples of challenging questions from the user profile samples that are created after anomaly detection analyzation has been done on Big Data.

6.2 Future Works

One of the future works is that implementing more combinations of models can be useful with increasing the data dimensions. Secondly, provide an algorithm to create the user profile database from the anomaly detection results. Also, implementing an algorithm to create the security questions automatically from the user profile database. As a result of this thesis, measuring human dynamics for next generation authentication and FictiZon collects a lot of real-time information about their subscribers are very important future works.

Furthermore, development of a novel Big Data-driven authentication as a service model and development of an integration framework to facilitate the collaboration and interoperability of multiple Big Data-driven authentication service providers are future works in this research. These two important future works can be done with these tasks: 1) design and develop SaaS-based authentication model (AUTHaaS), 2) a new integration framework will be designed and developed (iAUTH) in order to facilitate the collaboration and interoperability among multiple AUTHaaS providers.

This thesis is part of a research that providing new use cases for businesses seeking strong authentication and high market reputation. It also will help businesses to give their clients the sense of real security and to gain their admirations as a reward for protecting their assets.

References

[1] A. Ouda, "A Framework for Next Generation User Authentication," in 3rd MEC International Conference on Big Data and Smart City, Muscat, 2016.

[2] V. Chandola, A. Banerjee, V. Kumar, "Anomaly detection: A survey," ACM Computing Surveys (CSUR), vol. 41, no. 3, pp. 1-58, 2009.

[3] M. Ahmed, A. N. Mahmood and J. Hu, "A survey of network anomaly detection technique," Journal of Network and Computer Applications, vol. 60, pp. 19-31, 2016.

[4] M. H. Bhuyan, D. K. Bhattacharyya and J. K. Kalita, "Network Anomaly Detection: Methods, Systems and Tools," IEEE Communications Surveys & Tutorials, vol. 16, no. 1, pp. 303-336, 2014.

[5] V. Bontupalli and T. M. Taha, "Comprehensive Survey on Intrusion Detection on various hardware and software," in 2015 National Aerospace and Electronics Conference, Dayton, OH, USA, 2015.

[6] A. I. Rana, G. Estrada, M. Sole and V. Muntes, "Anomaly Detection Guidelines for Data Streams in Big Data," in 2016 3rd International Conference on Soft Computing & Machine Intelligence (ISCMI), 2016.

[7] H. S. Wu, "A survey of research on anomaly detection for time series," 13th International Computer Conference on Wavelet Active Media Technology and Information Processing, Chengdu, 2016.

[8] N. Patil and P. K. Biswas, "A survey of video datasets for anomaly detection in automated surveillance," Sixth International Symposium on Embedded Computing and System Design (ISED), Patna, 2016.

[9] K. Anand, J. Kumar and K. Anand, "Anomaly detection in online social network: A survey," International Conference on Inventive Communication and Computational Technologies, Coimbatore, 2017.

[10] B. Al-Musawi, P. Branch and G. Armitage, "BGP Anomaly Detection Techniques: A Survey," IEEE Communications Surveys & Tutorials, vol. 19, no. 1, pp. 377-396, 2017.

[11] R. Kaur and S. Singh, "A survey of data mining and social network analysis based anomaly detection techniques," Egyptian Informatics Journal, vol. 17, no. 2, pp. 199-216, 2016.

[12] H. Fanaee and J. Gama, "Tensor-based anomaly detection: An interdisciplinary survey," Knowledge-Based Systems, vol. 98, pp. 130-147, 2016.

[13] R. S. a. Y. Z. C. Gupta, "Eagle: User profile-based anomaly detection for securing hadoop clusters," in IEEE International Conference on Big Data, 2015.

[14] I. I. M. A. S. a. A. Ouda, "Data Analytics Methods for Anomaly Detection: Evolution and Recommendations," in International Conference for Digital Processing and Information Security, Dubai, 2018.

[15] S. Mehnaz and E. Bertino, "Building robust temporal user profiles for anomaly detection in file system accesses," in 2016 14th Annual Conference on Privacy, Security and Trust (PST), Auckland, 2016.

[16] J. Henriques et al., "Outliers detection in network services with self-learned profiles," in 2017 9th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT), Munich, 2017.

[17] A. T. A. O. K. O. a. A. K. C. T. Tiwari, "User-profile-based analytics for detecting cloud security breaches," in 2017 IEEE International Conference on Big Data (Big Data), Boston, MA, 2017.

[18] H. C. a. Z. Q. Y. Wang, "The design of database anomalous detection model based on user behaviour profile mining,"," in 3rd International Conference on Computer Science and Information Technology, Chengdu,, 2010.

[19] R. N. a. P. P. S. R. Ramachandran, "Anomaly Detection in Role Administered Relational Databases — A Novel Method," in 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Bangalore, India, 2018.

[20] G. P. a. H. Ashman, "Anomaly Detection over User Profiles for Intrusion," in Australian Information Security Management, Perth Western Australia, 2010.

[21] G. M. a. A. C. Malcolm Corney, "Detection of Anomalies from User Profiles Generated from System Logs," in Australasian Information Security Conference, Perth, Australia, 20111.

[22] M. C. a. A. A. G. Arash Habibi Lashkari, "A Survey on User Profiling Model for," Journal of Cyber Security and Mobility, vol. Vol. 8 , no. 1, p. 75–112., 2018.

[23] R. Jeyauthmigha and R. Suganthe, "Recursive Feature Elimination and Clustering Technique for Network Anomaly Detection," in 2018 International Conference on Current Trends towards Converging Technologies (ICCTCT), Coimbatore, 2018.

[24] M. Ahmed, "Thwarting DoS Attacks: A Framework for Detection based on Collective Anomalies and Clustering," Computer, vol. 50 , no. 9, pp. 76-82 , 2017.

[25] D. Iyer, A. Mohanpurkar, S. Janardhan, D. Rathod and A. Sardeshmukh, "Credit card fraud detection using Hidden Markov Model," 2011 World Congress on Information and Communication Technologies, Mumbai, 2011, pp. 1062-1066. doi: 10.1109/WICT.2011.6141395

[26] H. Zhu, Y. Xin and F. Wang, "A novel framework for anomaly detection based on hybrid HMM-SVM model," 2011 4th IEEE International Conference on Broadband Network and Multimedia Technology, Shenzhen, 2011, pp. 670-674. doi: 10.1109/ICBNMT.2011.6156020

[27] M. H. Rahmani and F. Almasganj, "Lip-reading via a DNN-HMM hybrid system using combination of the image-based and model-based features," 2017 3rd International Conference on Pattern Recognition and Image Analysis (IPRIA), Shahrekord, 2017, pp. 195-199. doi: 10.1109/PRIA.2017.7983045

[28] X. Wang, H. Wu and Z. Yi, "Research on Bank Anti-Fraud Model Based on K-means and Hidden Markov Model," 2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC), Chongqing, 2018, pp. 780-784. doi: 10.1109/ICIVC.2018.8492795

[29] A. K. Jain, S. S. Ahmed, P. Sundaramoorthy, R. Thiruvengadam and V. Vijayaraghavan, "Current peak based device classification in NILM on a low-cost embedded platform using extra-trees," 2017 IEEE MIT Undergraduate Research Technology Conference (URTC), Cambridge, MA, 2017, pp. 1-4. doi: 10.1109/URTC.2017.8284200.

[30] L. Han, "Research of K-MEANS Algorithm Based on Information Entropy in Anomaly Detection," 2012 Fourth International Conference on Multimedia Information Networking and Security, Nanjing, 2012, pp. 71-74. doi: 10.1109/MINES.2012.169.

[31] L. Han, "Using a Dynamic K-means Algorithm to Detect Anomaly Activities," 2011 Seventh International Conference on Computational Intelligence and Security, Hainan, 2011, pp. 1049-1052. doi: 10.1109/CIS.2011.233.

[32] P. Wang, L. Shi, B. Wang, Y. Wu and Y. Liu, "Survey on HMM based anomaly intrusion detection using system calls," 2010 5th International Conference on Computer Science & Education, Hefei, 2010, pp. 102-105. doi: 10.1109/ICCSE.2010.5593839.

[33] F. Farahnakian and J. Heikkonen, "A deep auto-encoder based approach for intrusion detection system," 2018 20th International Conference on Advanced Communication Technology (ICACT), Chuncheon-si Gangwon-do, Korea (South), 2018, pp. 178-183. doi: 10.23919/ICACT.2018.8323688.

[34] Ashfield, J.M. and Shroyer, D.C. and McConnell, E.C. (2014), "Dynamic authentication engine". US Patent 8,745,698.

[35] A. Ibrahim and A. Ouda, "A Hybrid-based Filtering Approach for User Authentication," in 2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE), Windsor, 2017.

[36] A. Javed, D. Bletgen, F. Kohlar, M. Dürmuth and J. Schwenk, "Secure Fallback Authentication and the Trusted Friend Attack," 2014 IEEE 34th International Conference on Distributed Computing Systems Workshops (ICDCSW), Madrid, 2014, pp. 22-28. doi: 10.1109/ICDCSW.2014.30

[37] W. Anani and A. Ouda, "The importance of human dynamics in the future user authentication," 2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE), Windsor, ON, 2017, pp. 1-5. doi: 10.1109/CCECE.2017.7946790

[38] W. Khreich et al, "An anomaly detection system based on variable N-gram features and one-class SVM," Information and Software Technology, vol. 91, pp. 186-197, 2017.

[39] A. Ibrahim and A. Ouda, "Innovative Data Authentication Model," 2016 IEEE 7th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), Vancouver, BC, 2016, pp. 1-7. doi: 10.1109/IEMCON.2016.7746268.

[40] M. Usha and P. Kavitha, "Anomaly based intrusion detection for 802.11 networks with optimal features using SVM classifier," Wireless Networks, vol. 23, (8), pp. 2431-2446, 2017.

[41] H. Saeedi Emadi and S. M. Mazinani, "A Novel Anomaly Detection Algorithm Using DBSCAN and SVM in Wireless Sensor Networks," Wireless Personal Communications, vol. 98, (2), pp. 2025-2035, 2018.

[42] L. Marti et al, "On the combination of support vector machines and segmentation algorithms for anomaly detection: A petroleum industry comparative study," Journal of Applied Logic, vol. 24, pp. 71-84, 2017.

[43] W. D. Fisher, T. K. Camp and V. V. Krzhizhanovskaya, "Anomaly detection in earth dam and levee passive seismic data using support vector machines and automatic feature selection," Journal of Computational Science, vol. 20, pp. 143-153, 2017.

[44] Y. Chen and W. Wu, "Mapping mineral prospectivity by using one-class support vector machine to identify multivariate geological anomalies from digital geological survey data," Australian Journal of Earth Sciences, vol. 64, (5), pp. 639, 2017.

[45] R. R. Reddy, Y. Ramadevi and K. V. N. Sunitha, "Enhanced anomaly detection using ensemble support vector machine," 2017 International Conference on Big Data Analytics and Computational Intelligence (ICBDAC), Chirala, 2017, pp. 107-111.

[46] R. Marapareddy, J. Aanstoos, and N. Younan, 2017, "Accuracy Analysis Comparison of Supervised Classification Methods for Anomaly Detection on Levees Using SAR Imagery," Electronics, v. 6, pp. 83.

[47] W. Huang et al, "An Anomaly Detection Method Based on Normalized Mutual Information Feature Selection and Quantum Wavelet Neural Network," Wireless Personal Communications, vol. 96, (2), pp. 2693-2713, 2017.

[48] J. Goh et al, "Anomaly detection in cyber physical systems using recurrent neural networks," in 2017, DOI: 10.1109/HASE.2017.36.

[49] M. Sabokrou et al, "Deep-Cascade: Cascading 3D Deep Neural Networks for Fast Anomaly Detection and Localization in Crowded Scenes," IEEE Trans. on Image Processing, v. 26, pp. 1992-2004, 2017.

[50] T. Tanprasert, C. Saiprasert and S. Thajchayapong, "Combining Unsupervised Anomaly Detection and Neural Networks for Driver Identification," Journal of Advanced Transportation, vol. 2017, 2017.

[51] M. Wess, P. D. S. Manoj and A. Jantsch, "Neural network based ECG anomaly detection on FPGA and trade-off analysis," in 2017.

[52] S. Andropov et al, "Network anomaly detection using artificial neural networks," in 2017, DOI: 10.23919/FRUCT.2017.8071288.

[53] Gunawansyah, T. H. Liong and Adiwijaya, "Prediction and anomaly detection of rainfall using evolving neural network to support planting calendar in soreang (bandung)," in 2017.

[54] A Bochem, H Zhang, D. Hogrefe, "Poster abstract: Streamlined anomaly detection in web requests using recurrent neural networks," in 2017.

[55] L. Han, "Research of K-MEANS Algorithm Based on Information Entropy in Anomaly Detection," 2012 Fourth International Conference on Multimedia Information Networking and Security, Nanjing, 2012, pp. 71-74. doi: 10.1109/MINES.2012.169.

[56] L. Han, "Using a Dynamic K-means Algorithm to Detect Anomaly Activities," 2011 Seventh International Conference on Computational Intelligence and Security, Hainan, 2011, pp. 1049-1052. doi: 10.1109/CIS.2011.233.

[57] C. Yin, S. Zhang, J. Wang and J. Kim, "An Improved K-means Using in Anomaly Detection," 2015 First International Conference on Computational Intelligence Theory, Systems and Applications (CCITSA), Yilan, 2015, pp. 129-132.

[58] M. Eslamnezhad and A. Y. Varjani, "Intrusion detection based on MinMax K-means clustering," 7'th International Symposium on Telecommunications (IST'2014), Tehran, 2014, pp. 804-808. doi: 10.1109/ISTEL.2014.7000814.

[59] M. E. KarsligEl, A. G. Yavuz, M. A. Güvensan, K. Hanifi and H. Bank, "Network intrusion detection using machine learning anomaly detection algorithms," 2017 25th Signal Processing and Communications Applications Conference (SIU), Antalya, 2017, pp. 1-4. doi: 10.1109/SIU.2017.7960616.

[60] S. Varuna and P. Natesan, "An integration of K-means clustering and naïve bayes classifier for Intrusion Detection," 2015 3rd International Conference on Signal

Processing, Communication and Networking (ICSCN), Chennai, 2015, pp. 1-5. doi: 10.1109/ICSCN.2015.7219835

[61] X. Zhao and W. Zhang, "An Anomaly Intrusion Detection Method Based on Improved K-means of Cloud Computing," 2016 Sixth International Conference on Instrumentation & Measurement, Computer, Communication and Control (IMCCC), Harbin, 2016, pp. 284-288. doi: 10.1109/IMCCC.2016.108.

[62] D. M. Menon and N. Radhika, "Anomaly detection in smart grid traffic data for home area network," 2016 International Conference on Circuit, Power and Computing Technologies (ICCPCT), Nagercoil, 2016, pp. 1-4. doi: 10.1109/ICCPCT.2016.7530186.

[63] K. Tago and Q. Jin, "Detection of Anomaly Health Data by Specifying Latent Factors with SEM and Estimating Hidden States with HMM," 2018 9th International Conference on Information Technology in Medicine and Education (ITME), Hangzhou, 2018, pp. 137-141. doi: 10.1109/ITME.2018.00040.

[64] S. Poh, Y. Tan, X. Guo, S. Cheong, C. Ooi and W. Tan, "LSTM and HMM Comparison for Home Activity Anomaly Detection," 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC), Chengdu, China, 2019, pp. 1564-1568. doi: 10.1109/ITNEC.2019.8729168.

[65] F. Wang, H. Zhu, B. Tian, Y. Xin, X. Niu and Y. Yang, "A HMM-based method for anomaly detection," 2011 4th IEEE International Conference on Broadband Network and Multimedia Technology, Shenzhen, 2011, pp. 276-280. doi: 10.1109/ICBNMT.2011.6155940.

[66] H. Zhu, Y. Xin and F. Wang, "A novel framework for anomaly detection based on hybrid HMM-SVM model," 2011 4th IEEE International Conference on Broadband Network and Multimedia Technology, Shenzhen, 2011, pp. 670-674. doi: 10.1109/ICBNMT.2011.6156020.

[67] S. Alhaidari and M. Zohdy, "Network Anomaly Detection Using Two-Dimensional Hidden Markov Model Based Viterbi Algorithm," 2019 IEEE International Conference On Artificial Intelligence Testing (AITest), Newark, CA, USA, 2019, pp. 17-18. doi: 10.1109/AITest.2019.00-14.

[68] B. Lorbeer, T. Deutsch, P. Ruppel and A. Küpper, "Anomaly Detection with HMM Gauge Likelihood Analysis," 2019 IEEE Fifth International Conference on Big Data Computing Service and Applications (BigDataService), Newark, CA, USA, 2019, pp. 1-8. doi: 10.1109/BigDataService.2019.00008.

[69] Panhong Wang, Liang Shi, Beizhan Wang, Yangbin Liu and Yuanqin Wu, "A method for HMM-based system calls intrusion detection based on hybrid training algorithm," 2011 IEEE International Conference on Information and Automation, Shenzhen, 2011, pp. 339-342. doi: 10.1109/ICINFA.2011.5949013.

[70] W. Honghao, J. Yunfeng and W. Lei, "Spectrum anomalies autonomous detection in cognitive radio using Hidden Markov Models," 2015 IEEE Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), Chongqing, 2015, pp. 388-392. doi: 10.1109/IAEAC.2015.7428581.

[71] A. M. Vartouni, S. S. Kashi and M. Teshnehlab, "An anomaly detection method to detect web attacks using Stacked Auto-Encoder," 2018 6th Iranian Joint Congress on Fuzzy and Intelligent Systems (CFIS), Kerman, 2018, pp. 131-134. doi: 10.1109/CFIS.2018.8336654.

[72] Y. Cui, Y. Sun, J. Hu and G. Sheng, "A Convolutional Auto-Encoder Method for Anomaly Detection on System Logs," 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Miyazaki, Japan, 2018, pp. 3057-3062. doi: 10.1109/SMC.2018.00519.

[73] C. Aytekin, X. Ni, F. Cricri and E. Aksu, "Clustering and Unsupervised Anomaly Detection with l2Normalized Deep Auto-Encoder Representations," 2018 International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, 2018, pp. 1-6. doi: 10.1109/IJCNN.2018.8489068.

[74] J. Sun, X. Wang, N. Xiong and J. Shao, "Learning Sparse Representation With Variational Auto-Encoder for Anomaly Detection," in IEEE Access, vol. 6, pp. 33353-33361, 2018. doi: 10.1109/ACCESS.2018.2848210.

[75] M. Jeragh and M. AlSulaimi, "Combining Auto Encoders and One Class Support Vectors Machine for Fraudulent Credit Card Transactions Detection," 2018 Second World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4), London, 2018, pp. 178-184. doi: 10.1109/WorldS4.2018.8611624.

[76] M. Sabokrou, M. Fathy and M. Hoseini, "Video anomaly detection and localisation based on the sparsity and reconstruction error of auto-encoder," in Electronics Letters, vol. 52, no. 13, pp. 1122-1124, 23 6 2016. doi: 10.1049/el.2016.0440.

[77] H. Wang et al., "Optimization of Reconstruction Accuracy of Anomaly Position Based on Stacked Auto-Encoder Neural Networks," in IEEE Access, vol. 7, pp. 116578-116584, 2019. doi: 10.1109/ACCESS.2019.2931995.

[78] A. Hosseini and M. Sarrafzadeh, "Unsupervised Prediction of Negative Health Events Ahead of Time," 2019 IEEE EMBS International Conference on Biomedical & Health Informatics (BHI), Chicago, IL, USA, 2019, pp. 1-4. doi: 10.1109/BHI.2019.8834550.

[79] W. Cui and H. Wang, "Anomaly detection and visualization of school electricity consumption data," 2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA)(, Beijing, 2017, pp. 606-611. doi: 10.1109/ICBDA.2017.8078707.

[80] D. Ma, Y. Yuan and Q. Wang, "A sparse dictionary learning method for hyperspectral anomaly detection with capped norm," 2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Fort Worth, TX, 2017, pp. 648-651. doi: 10.1109/IGARSS.2017.8127037.

[81] A. Maurya and M. Cheung, "Contrastive Structured Anomaly Detection for Gaussian Graphical Models," 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM), Barcelona, 2018, pp. 736-739. doi: 10.1109/ASONAM.2018.8508475.

[82] S. Küçük and S. E. Yüksel, "Comparison of RX-based anomaly detectors on synthetic and real hyperspectral data," 2015 7th Workshop on Hyperspectral Image and Signal Processing: Evolution in Remote Sensing (WHISPERS), Tokyo, 2015, pp. 1-4. doi: 10.1109/WHISPERS.2015.8075504.

[83] H. Luo and S. Zhong, "Gas turbine engine gas path anomaly detection using deep learning with Gaussian distribution," 2017 Prognostics and System Health Management Conference (PHM-Harbin), Harbin, 2017, pp. 1-6. doi: 10.1109/PHM.2017.8079166.

[84] S. Fan, G. Liu and Z. Chen, "Anomaly detection methods for bankruptcy prediction," 2017 4th International Conference on Systems and Informatics (ICSAI), Hangzhou, 2017, pp. 1456-1460. doi: 10.1109/ICSAI.2017.8248515.

[85] M. Bahrololum and M. Khaleghi, "Anomaly Intrusion Detection System Using Gaussian Mixture Model," 2008 Third International Conference on Convergence and Hybrid Information Technology, Busan, 2008, pp. 1162-1167. doi: 10.1109/ICCIT.2008.17.

[86] J. Frontera-Pons, M. A. Veganzones, S. Velasco-Forero, F. Pascal, J. P. Ovarlez and J. Chanussot, "Robust anomaly detection in Hyperspectral Imaging," 2014 IEEE Geoscience and Remote Sensing Symposium, Quebec City, QC, 2014, pp. 4604-4607. doi: 10.1109/IGARSS.2014.6947518.

[87] S. Gaharana and D. Anand, "Dynamic Id Based Remote User Authentication in Multi Server Environment Using Smart Cards: A Review," 2015 International Conference on Computational Intelligence and Communication Networks (CICN), Jabalpur, 2015, pp. 1081-1084. doi: 10.1109/CICN.2015.212.

[88] X. Zhou, Y. Xiong, F. Miao and M. Li, "A new dynamic user authentication scheme using smart cards for wireless sensor network," 2011 IEEE 2nd International Conference on Computing, Control and Industrial Engineering, Wuhan, 2011, pp. 1-4. doi: 10.1109/CCIENG.2011.6008052.

[89] S. Shen, T. Kang, S. Lin and W. Chien, "Random graphic user password authentication scheme in mobile devices," 2017 International Conference on Applied System Innovation (ICASI), Sapporo, 2017, pp. 1251-1254. doi: 10.1109/ICASI.2017.7988123.

[90] L. Pan and S. Bangay, "Generating Reputable, Memorizable, and Privacy Preserving Security Questions Using the Propp Theory of Narrative," 2014 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery, Shanghai, 2014, pp. 66-72. doi: 10.1109/CyberC.2014.20.

[91] N. Micallef and N. A. G. Arachchilage, "Changing users' security behaviour towards security questions: A game based learning approach," 2017 Military Communications and Information Systems Conference (MilCIS), Canberra, ACT, 2017, pp. 1-6. doi: 10.1109/MilCIS.2017.8190424.

[92] A. Hang, A. De Luca and H. Hussmann, "I know what you did last week! do you?: Dynamic security questions for fallback authentication on smartphones," Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems. ACM, 2015.

[93] S. N. Basharzad and M. Fazeli, "Knowledge based dynamic password," 2017 IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI), Tehran, 2017, pp. 0367-0372. doi: 10.1109/KBEI.2017.8325004.

[94] I. I. M. Abu Sulayman and A. Ouda, "User Modeling via Anomaly Detection Techniques for User Authentication," 2019 10th Annual Information Technology, Electronics and Mobile Communication (IEMCON), VANCOUVER, Canada, 2019.

[95] C. Lile and L. Yiqun, "Anomaly detection in thermal images using deep neural networks," in 2017, . DOI: 10.1109/ICIP.2017.8296692.

[96] P. Napoletano, F. Piccoli and R. Schettini, "Anomaly Detection in Nanofibrous Materials by CNN-Based Self-Similarity," Sensors, vol. 18, (1), pp. 209, 2018.

[97] C. Ting, R. Field, A. Fisher and T. Bauer, "Compression Analytics for Classification and Anomaly Detection Within Network Communication," in IEEE Transactions on Information Forensics and Security, vol. 14, no. 5, pp. 1366-1376, May 2019. doi: 10.1109/TIFS.2018.2878172

[98] M. Kallenberg et al., "Unsupervised Deep Learning Applied to Breast Density Segmentation and Mammographic Risk Scoring," in IEEE Transactions on Medical Imaging, vol. 35, no. 5, pp. 1322-1331, May 2016. doi: 10.1109/TMI.2016.2532122

[99] Eskin E., Arnold A., Prerau M., Portnoy L., Stolfo S. (2002) A Geometric Framework for Unsupervised Anomaly Detection. In: Barbará D., Jajodia S. (eds) Applications of Data Mining in Computer Security. Advances in Information Security, vol 6. Springer, Boston, MA

[100] Y. Wang et al., "Iterative anomaly detection," 2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS), Fort Worth, TX, 2017, pp. 586-589. doi: 10.1109/IGARSS.2017.8127021

[101] S. Asanger and A. Hutchison, "Experiences and Challenges in Enhancing Security Information and Event Management Capability Using Unsupervised Anomaly

Detection," in International Conference on Availability, Reliability and Security, Regensburg, 2013.

[102] M. H. A. a. M. P. Jing Tian, "Anomaly Detection Using Self-Organizing Maps-Based K-Nearest Neighbour Algorithm," in EUROPEAN CONFERENCE OF THE PROGNOSTICS AND HEALTH MANAGEMENT SOCIETY, 2014.

[103] a. C. B. Gaby Abou Haidar, "High Perception Intrusion Detection Systems Using," in 2015 Ninth International Conference on Complex, Intelligent, and Software Intensive Systems, 2015.

[104] S.-Y. Huang and Y.-N. Huang, "Network traffic anomaly detection based on growing," in 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN), Budapest, 2013.

[105] "Network forensic analysis using growing," in 2013 IEEE 13th International Conference on Data Mining Workshops, Dallas, TX, 2013.

[106] Qian Gao, Su Mei Xi and Young Im Cho, "A multi-agent personalized ontology profile based user preference profile construction method," IEEE ISR 2013, Seoul, 2013, pp. 1-4. doi: 10.1109/ISR.2013.6695695.

[107] S. Hu, Z. Xiao, Q. Rao and R. Liao, "An anomaly detection model of user behavior based on similarity clustering," 2018 IEEE 4th Information Technology and Mechatronics Engineering Conference (ITOEC), Chongqing, China, 2018, pp. 835-838. doi: 10.1109/ITOEC.2018.8740748.

[108] R. Ramachandran, R. Nidhin and P. P. Shogil, "Anomaly Detection in Role Administered Relational Databases — A Novel Method," 2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI), Bangalore, 2018, pp. 1017-1021. doi: 10.1109/ICACCI.2018.8554752.

[109] J. Y. Kim and R. E. Gantenbein, "Automated Anomaly Detection Using Time-Variant Normal Profiling," 2006 World Automation Congress, Budapest, 2006, pp. 1-4. doi: 10.1109/WAC.2006.376026.

[110] I. Bae, H. Lee and K. Lee, "Design and Evaluation of a Dynamic Anomaly Detection Scheme Using the Age of User Profiles," Fourth International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007), Haikou, 2007, pp. 136-140. doi: 10.1109/FSKD.2007.241.

[111] C. Gupta, R. Sinha and Y. Zhang, "Eagle: User profile-based anomaly detection for securing Hadoop clusters," 2015 IEEE International Conference on Big Data (Big Data), Santa Clara, CA, 2015, pp. 1336-1343. doi: 10.1109/BigData.2015.7363892.

[112] Yaohui Wang, Hongjian Chu and Zhaoyang Qu, "The design of database anomalous detection model based on user behavior profile mining," 2010 3rd International Conference on Computer Science and Information Technology, Chengdu, 2010, pp. 472-475. doi: 10.1109/ICCSIT.2010.5564082.

[113] Z. Malek and B. Trivedi, "GUI-based user behavior intrusion detection," 2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI), Chennai, 2017, pp. 2050-2055. doi: 10.1109/ICPCSI.2017.8392076.

[114] T. Ait Tchakoucht, M. Ezziyyani, M. Jbilou and M. Salaun, "Behavioral appraoch for intrusion detection," 2015 IEEE/ACS 12th International Conference of Computer Systems and Applications (AICCSA), Marrakech, 2015, pp. 1-5. doi: 10.1109/AICCSA.2015.7507118.

[115] M. Singh, B. M. Mehtre and S. Sangeetha, "User Behavior Profiling using Ensemble Approach for Insider Threat Detection," 2019 IEEE 5th International Conference on Identity, Security, and Behavior Analysis (ISBA), Hyderabad, India, 2019, pp. 1-8. doi: 10.1109/ISBA.2019.8778466.

[116] S. Batthalla, M. Swarnkar, N. Hubballi and M. Natu, "VoIP Profiler: Profiling Voice over IP User Communication Behavior," 2016 11th International Conference on Availability, Reliability and Security (ARES), Salzburg, 2016, pp. 312-320. doi: 10.1109/ARES.2016.19.

Appendices

Appendix A: All the results for all data in assumption 1 (without normalization or dimensional reduction)

Credit Card Dataset										
Models	Accuracy	Precision	Recall	F1-score	ROC auc score	RMSE	TPR	TNR	FPR	FNR
K-means	0.5329	0.4988	0.4315	0.3504	0.4315	0.6834	0.3293	0.5338	0.4662	0.6707
HMM	0.8431	0.512	0.8767	0.4811	0.8767	0.3961	0.9106	0.8428	0.1572	0.0894
Auto-encoder	0.0043	0.0022	0.5	0.0043	0.5	0.9978	1	0	1	0
Gaussian	0.9889	0.5633	0.6342	0.5855	0.6342	0.1054	0.2764	0.992	0.008	0.7236
Synthetic Dataset										
K-means	0.4914	0.4971	0.4747	0.3517	0.4747	0.7132	0.4569	0.4924	0.5076	0.5431
HMM	0.1368	0.4107	0.0705	0.1203	0.0705	0.9291	0	0.141	0.859	1
Auto-encoder	0.0297	0.0149	0.5	0.0289	0.5	0.985	1	0	1	0
Gaussian	0.9862	0.9532	0.7932	0.855	0.7932	0.1174	0.5881	0.9984	0.0016	0.4119
Germen Dataset										
K-means	0.2366	0.4704	0.47	0.2366	0.47	0.8737	0.14	0.8	0.2	0.86
HMM	0.3073	0.519	0.5252	0.3045	0.5252	0.8323	0.2171	0.8333	0.1667	0.7829
Auto-encoder	0.8537	0.4268	0.5	0.4605	0.5	0.3825	1	0	1	0
Gaussian	0.1976	0.5189	0.5093	0.1923	0.5093	0.8958	0.0686	0.95	0.05	0.9314
small server computer Dataset										
K-means	0.4886	0.5024	0.5211	0.3543	0.5211	0.7151	0.5556	0.4866	0.5134	0.4444
HMM	0.9902	0.9342	0.8872	0.9093	0.8872	0.0989	0.7778	0.9966	0.0034	0.2222

Auto-encoder	0.0293	0.0147	0.5	0.0285	0.5	0.9852	1	0	1	0
Gaussian	0.9772	0.9885	0.6111	0.676	0.6111	0.151	0.2222	1	0	0.7778
High dimensional server computer Dataset										
K-means	0.52	0.4548	0.3778	0.3763	0.3778	0.6928	0.2	0.5556	0.4444	0.8
HMM	0.54	0.4919	0.4778	0.4165	0.4778	0.6782	0.4	0.5556	0.4444	0.6
Auto-encoder	0.1	0.05	0.5	0.0909	0.5	0.9487	0	1	0	
Gaussian	0.92	0.9592	0.6	0.6454	0.6	0.2828	0.2	1	0	0.8
eecs498 Dataset										
K-means	0.4987	0.4995	0.4865	0.3406	0.4865	0.7081	0.474	0.4989	0.5011	0.526
HMM	0.9862	0.7048	0.993	0.7871	0.993	0.1175	1	0.9861	0.0139	0
Auto-encoder	0.0096	0.0048	0.5	0.0095	0.5	0.9952	1	0	1	0
Gaussian	0.9982	0.9208	0.9991	0.9565	0.9991	0.0425	1	0.9982	0.0018	0
Porto Seguro’s Safe Driver Prediction Dataset										
K-means	0.5002	0.5008	0.5025	0.3975	0.5025	0.707	0.5054	0.4997	0.5003	0.4946
HMM	0.1013	0.4884	0.4973	0.0972	0.4973	0.948	0.9759	0.0186	0.9814	0.0241
Auto-encoder	0.0864	0.0432	0.5	0.0795	0.5	0.9558	1	0	1	0
Gaussian	0.9136	0.4568	0.5	0.4774	0.5	0.2939	0	1	0	1
santander-customer-transaction Dataset										
K-means	0.5052	0.5046	0.5068	0.4622	0.5068	0.7034	0.5097	0.5039	0.4961	0.4903
HMM	0.4947	0.4954	0.4932	0.4516	0.4932	0.7108	0.4906	0.4959	0.5041	0.5094
Auto-encoder	0.2183	0.1092	0.5	0.1792	0.5	0.8841	1	0	1	0
Gaussian	0.21832	0.10916	0.5	0.1792	0.5	0.88413	1	0	1	0

Prudential Life Insurance Assessment dataset										
K-means	0.5037	0.5037	0.5037	0.503	0.5037	0.7045	0.5041	0.5034	0.4966	0.4959
HMM	0.3825	0.3833	0.393	0.3748	0.393	0.7858	0.5332	0.2528	0.7472	0.4668
Auto-encoder	0.4624	0.2312	0.5	0.3162	0.5	0.7332	1	0	1	0
Gaussian	0.4624	0.2312	0.5	0.3162	0.5	0.7332	1	0	1	0

Appendix B: All the results for all data in assumption 2 (with normalization only)

Credit Card Dataset										
Models	Accuracy	Precision	Recall	F1-score	ROC auc score	RMSE	TPR	TNR	FPR	FNR
K-means	0.5256	0.4983	0.4036	0.3468	0.4036	0.6888	0.2805	0.5266	0.4734	0.7195
HMM	0.8432	0.512	0.8767	0.4811	0.8767	0.396	0.9106	0.8429	0.1571	0.0894
Auto-encoder	0.9816	0.5705	0.8167	0.6122	0.8167	0.1355	0.6504	0.9831	0.0169	0.3496
Gaussian	0.9921	0.6654	0.903	0.7336	0.903	0.0887	0.813	0.9929	0.0071	0.187
Synthetic Dataset										
K-means	0.114	0.425	0.2168	0.106	0.2168	0.9413	0.3261	0.1075	0.8925	0.6739
HMM	0.8632	0.5893	0.9295	0.6136	0.9295	0.3698	1	0.859	0.141	0
Auto-encoder	0.9762	0.9794	0.6018	0.6626	0.6018	0.1542	0.2036	0.9999	0.0001	0.7964
Gaussian	0.9843	0.9775	0.7427	0.8197	0.7427	0.1254	0.4858	0.9995	0.0005	0.5142
Dataset										
K-means	0.3	0.4599	0.4381	0.2907	0.4381	0.8367	0.2429	0.6333	0.3667	0.7571
HMM	0.3415	0.5306	0.5452	0.3351	0.5452	0.8115	0.2571	0.8333	0.1667	0.7429
Auto-encoder	0.1463	0.0732	0.5	0.1277	0.5	0.9239	0	1	0	1
Gaussian	0.2	0.521	0.5107	0.1951	0.5107	0.8944	0.0714	0.95	0.05	0.9286
small server computer Dataset										
K-means	0.5016	0.5093	0.5817	0.366	0.5817	0.706	0.6667	0.4966	0.5034	0.3333

HMM	0.0098	0.0658	0.1128	0.0098	0.1128	0.9951	0.2222	0.0034	0.9966	0.7778
Auto-encoder	0.9902	0.9342	0.8872	0.9093	0.8872	0.0989	0.7778	0.9966	0.0034	0.2222
Gaussian	0.9772	0.9885	0.6111	0.676	0.6111	0.151	0.2222	1	0	0.7778
High dimensional server computer Dataset										
K-means	0.59	0.5347	0.5944	0.4738	0.5944	0.6403	0.6	0.5889	0.4111	0.4
HMM	0.51	0.502	0.50556	0.41099	0.50556	0.7	0.5	0.51111	0.48889	0.5
Auto-encoder	0.9	0.45	0.5	0.4737	0.5	0.3162	0	1	0	1
Gaussian	0.9	0.45	0.5	0.4737	0.5	0.3162	0	1	0	1
eecs498 Dataset										
K-means	0.4987	0.4995	0.4865	0.3406	0.4865	0.708	0.474	0.499	0.501	0.526
HMM	0.9807	0.6659	0.9903	0.7443	0.9903	0.1388	1	0.9806	0.0194	0
Auto-encoder	0.9959	0.85	0.9979	0.9107	0.9979	0.064	1	0.9959	0.0041	0
Gaussian	0.9983	0.9254	0.9992	0.9593	0.9992	0.041	1	0.9983	0.0017	0
Porto Seguro’s Safe Driver Prediction Dataset										
K-means	0.3133	0.4756	0.4352	0.2808	0.4352	0.8287	0.5825	0.2878	0.7122	0.4175
HMM	0.8528	0.5214	0.5202	0.5207	0.5202	0.3837	0.1182	0.9223	0.0777	0.8818
Auto-encoder	0.9115	0.5162	0.5006	0.4806	0.5006	0.2976	0.0039	0.9973	0.0027	0.9961
Gaussian	0.0864	0.0432	0.5	0.0795	0.5	0.9558	1	0	1	0
santander-customer-transaction Dataset										
K-means	0.4999	0.4969	0.4955	0.455	0.4955	0.7072	0.4876	0.5034	0.4966	0.5124
HMM	0.5145	0.5176	0.5259	0.4745	0.5259	0.6967	0.5459	0.5058	0.4942	0.4541
Auto-encoder	0.7817	0.3908	0.5	0.4387	0.5	0.4672	0	1	0	1
Gaussian	0.2183	0.1092	0.5	0.1792	0.5	0.8841	1	0	1	0
Prudential Life Insurance Assessment dataset										
K-means	0.5169	0.5114	0.5112	0.5105	0.5112	0.6951	0.4361	0.5864	0.4136	0.5639
HMM	0.4836	0.4895	0.4897	0.4826	0.4897	0.7186	0.5709	0.4084	0.5916	0.4291

Auto-encoder	0.5467	0.6486	0.5105	0.3789	0.5105	0.6733	0.0291	0.9919	0.0081	0.9709
Gaussian	0.622	0.6428	0.633	0.6184	0.633	0.6148	0.7789	0.4871	0.5129	0.2211

Appendix C: All the results for all data in assumption 3 (with dimensional reduction only)

Credit Card Dataset										
Models	Accuracy	Precision	Recall	F1-score	ROC auc score	RMSE	TPR	TNR	FPR	FNR
K-means	0.5329	0.4988	0.4315	0.3504	0.4315	0.6834	0.3293	0.533792	0.4662	0.6707
HMM	0.7751	0.5075	0.8061	0.4519	0.8061	0.4743	0.8374	0.774792	0.2252	0.1626
Auto-encoder	0.0043	0.0022	0.5	0.0043	0.5	0.9978	1	0	1	0
Gaussian	0.9946	0.6528	0.6027	0.6227	0.6027	0.0736	0.2073	0.997995	0.002	0.7927
Synthetic Dataset										
K-means	0.5084	0.5029	0.5252	0.3643	0.5252	0.7012	0.5431	0.50729	0.4927	0.4569
HMM	0.8834	0.5933	0.8897	0.625	0.8897	0.3414	0.8964	0.883044	0.117	0.1036
Auto-encoder	0.0297	0.5149	0.5	0.0289	0.5	0.985	1	8.51E-06	1	0
Gaussian	0.985	0.9689	0.76	0.8328	0.76	0.1226	0.5208	0.9992	0.0008	0.4792
Dataset										
K-means	0.2366	0.4704	0.47	0.2366	0.47	0.8737	0.14	0.8	0.2	0.86
HMM	0.3537	0.5025	0.504	0.34	0.504	0.804	0.2914	0.716667	0.2833	0.7086
Auto-encoder	0.839	0.4257	0.4914	0.4562	0.4914	0.4012	0.9829	0	1	0.0171
Gaussian	0.1463	0.0732	0.5	0.1277	0.5	0.9239	0	1	0	1
small server computer Dataset										
K-means	0.3974	0.4969	0.4741	0.3049	0.4741	0.7763	0.5556	0.392617	0.6074	0.4444
HMM	0.013	0.0764	0.1683	0.013	0.1683	0.9935	0.3333	0.003356	0.9966	0.6667
Auto-encoder	0.9674	0.7257	0.8216	0.7643	0.8216	0.1805	0.6667	0.97651	0.0235	0.3333
Gaussian	0.9739	0.9869	0.5556	0.5934	0.5556	0.1614	0.1111	1	0	0.8889
High dimensional server computer Dataset										

K-means	0.56	0.5282	0.5778	0.4544	0.5778	0.6633	0.6	0.555556	0.4444	0.4
HMM	0.48	0.496	0.4889	0.3922	0.4889	0.7211	0.5	0.477778	0.5222	0.5
Auto-encoder	0.13	0.5515	0.5167	0.1257	0.5167	0.9327	1	0.033333	0.9667	0
Gaussian	0.9	0.45	0.5	0.4737	0.5	0.3162	0	1	0	1
eecs498 Dataset										
K-means	0.5011	0.5005	0.5134	0.3426	0.5134	0.7063	0.526	0.50091	0.4991	0.474
HMM	0.4987	0.4995	0.4865	0.3406	0.4865	0.708	0.474	0.498965	0.501	0.526
Auto-encoder	0.0098	0.5048	0.5001	0.0097	0.5001	0.9951	1	0.000188	0.9998	0
Gaussian	0.9904	0.4952	0.5	0.4976	0.5	0.0978	0	1	0	1
Porto Seguro’s Safe Driver Prediction Dataset										
K-means	0.5008	0.5008	0.5026	0.3978	0.5026	0.7065	0.5047	0.500453	0.4995	0.4953
HMM	0.1832	0.4777	0.4683	0.1817	0.4683	0.9038	0.8129	0.123649	0.8764	0.1871
Auto-encoder	0.0864	0.0432	0.5	0.0795	0.5	0.9558	1	0	1	0
Gaussian	0.9136	0.4568	0.5	0.4774	0.5	0.2939	0	1	0	1
santander-customer-transaction Dataset										
K-means	0.5052	0.5047	0.5068	0.4622	0.5068	0.7034	0.5097	0.503947	0.4961	0.4903
HMM	0.4946	0.4953	0.4932	0.4515	0.4932	0.7109	0.4906	0.49572	0.5043	0.5094
Auto-encoder	0.2183	0.1092	0.5	0.1792	0.5	0.8841	1	0	1	0
Gaussian	0.2183	0.1092	0.5	0.1792	0.5	0.8841	1	0	1	0
Prudential Life Insurance Assessment dataset										
K-means	0.5037	0.5037	0.5037	0.503	0.5037	0.7045	0.5041	0.503395	0.4966	0.4959
HMM	0.6267	0.6354	0.6331	0.6262	0.6331	0.611	0.718	0.548212	0.4518	0.282
Auto-encoder	0.4623	0.2312	0.4999	0.3162	0.4999	0.7333	0.9999	0	1	0.0001
Gaussian	0.4624	0.2312	0.5	0.3162	0.5	0.7332	1	0	1	0

Appendix D: All the results for all data in assumption 4 (with both normalization dimensional reduction only)

Credit Card Dataset										
Models	Accuracy	Precision	Recall	F1-score	ROC auc score	RMSE	TPR	TNR	FPR	FNR
K-means	0.4745	0.5017	0.5965	0.3269	0.5965	0.7249	0.7195	0.4734	0.526581	0.2805
HMM	0.1568	0.488	0.1233	0.1358	0.1233	0.9182	0.0894	0.1571	0.842886	0.9106
Auto-encoder	0.9831	0.5027	0.5079	0.5029	0.5079	0.1299	0.0285	0.9873	0.01275	0.9715
RNN										
Gaussian	0.9923	0.6674	0.903	0.7356	0.903	0.088	0.813	0.993	0.006964	0.187
Synthetic Dataset										
K-means	0.5845	0.5189	0.6608	0.4131	0.6608	0.6446	0.7419	0.5796	0.420354	0.2581
HMM	0.55	0.5265	0.7295	0.4037	0.7295	0.6708	0.9203	0.5386	0.461354	0.0797
Auto-encoder	0.9777	0.9718	0.629	0.6981	0.629	0.1494	0.2583	0.9997	0.000281	0.7417
Gaussian	0.9718	0.9833	0.5264	0.543	0.5264	0.1678	0.0528	1	8.51E-06	0.9472
Germen Dataset										
K-means	0.3024	0.4651	0.4464	0.2935	0.4464	0.8352	0.2429	0.65	0.35	0.7571
HMM	0.4268	0.5215	0.54	0.3971	0.54	0.7571	0.38	0.7	0.3	0.62
Auto-encoder	0.1732	0.5358	0.5088	0.1622	0.5088	0.9093	0.0343	0.9833	0.016667	0.9657
Gaussian	0.1488	0.5733	0.5014	0.1308	0.5014	0.9226	0.0029	1	0	0.9971
small server computer Dataset										
K-means	0.3648	0.4947	0.4573	0.286	0.4573	0.797	0.5556	0.3591	0.64094	0.4444
HMM	0.9902	0.995	0.8333	0.8975	0.8333	0.0989	0.6667	1	0	0.3333
Auto-encoder	0.9902	0.995	0.8333	0.8975	0.8333	0.0989	0.6667	1	0	0.3333

Gaussian	0.9707	0.4853	0.5	0.4926	0.5	0.1712	0	1	0	1
High dimensional server computer Dataset										
K-means	0.58	0.5164	0.5444	0.4543	0.5444	0.6481	0.5	0.5889	0.411111	0.5
HMM	0.57	0.5303	0.5833	0.4608	0.5833	0.6557	0.6	0.5667	0.433333	0.4
Auto-encoder	0.91	0.7535	0.6833	0.7107	0.6833	0.3	0.4	0.9667	0.033333	0.6
Gaussian	0.9	0.45	0.5	0.4737	0.5	0.3162	0	1	0	1
eecs498 Dataset										
K-means	0.9883	0.4952	0.4989	0.4971	0.4989	0.1081	0	0.9979	0.002133	1
HMM	0.9791	0.6571	0.9895	0.7338	0.9895	0.1445	1	0.9789	0.02108	0
Auto-encoder	0.9966	0.8702	0.9983	0.9246	0.9983	0.0579	1	0.9966	0.003388	0
Gaussian	0.998	0.9157	0.9958	0.9522	0.9958	0.0446	0.9935	0.9981	0.001945	0.0065
Porto Seguro’s Safe Driver Prediction Dataset										
K-means	0.6828	0.5245	0.5657	0.4953	0.5657	0.5632	0.4241	0.7072	0.292754	0.5759
HMM	0.8657	0.5256	0.5198	0.5215	0.5198	0.3664	0.1017	0.938	0.062003	0.8983
Auto-encoder	0.9081	0.5256	0.5024	0.487	0.5024	0.3032	0.0121	0.9928	0.007184	0.9879
Gaussian	0.8944	0.5175	0.5055	0.4994	0.5055	0.325	0.0354	0.9756	0.024419	0.9646
santander-customer-transaction Dataset										
K-means	0.491	0.4891	0.484	0.4459	0.484	0.7135	0.4716	0.4964	0.503641	0.5284
HMM	0.5054	0.5046	0.5067	0.4623	0.5067	0.7033	0.509	0.5044	0.495636	0.491
Auto-encoder	0.2183	0.1092	0.5	0.1792	0.5	0.8841	1	0	1	0
Gaussian	0.2183	0.1092	0.5	0.1792	0.5	0.8841	1	0	1	0
Prudential Life Insurance Assessment dataset										
K-means	0.4831	0.4886	0.4888	0.4824	0.4888	0.7189	0.5639	0.4136	0.586351	0.4361

HMM	0.6077	0.6096	0.61	0.6076	0.61	0.6263	0.6404	0.5796	0.420439	0.3596
Auto-encoder	0.5564	0.6492	0.5218	0.4086	0.5218	0.666	0.0611	0.9825	0.017542	0.9389
Gaussian	0.8862	0.9012	0.8941	0.886	0.8941	0.3374	1	0.7883	0.211747	0

Appendix E: PCA comparison based on features for experiment 1.

Dataset 1										
Models	Accuracy	Precision	Recall	F1-score	ROC auc score	RMSE	TPR	TNR	FPR	FNR
PCA = 1	0.5329	0.4988	0.4315	0.3504	0.4315	0.6834	0.3293	0.5338	0.4662	0.6707
PCA = 2	0.5749	0.5038	0.7197	0.373	0.7197	0.652	0.8659	0.5736	0.4264	0.1341
PCA = 3	0.4004	0.4959	0.2719	0.2868	0.2719	0.7743	0.1423	0.4016	0.5984	0.8577
PCA = 4	0.4099	0.4963	0.2908	0.2917	0.2908	0.7682	0.1707	0.4109	0.5891	0.8293
PCA = 5	0.4671	0.5012	0.5685	0.323	0.5685	0.73	0.6707	0.4662	0.5338	0.3293
PCA = 6	0.53291	0.49882	0.43153	0.35038	0.43153	0.68344	0.32927	0.53379	0.46621	0.67073
PCA = 7	0.5329	0.4988	0.4315	0.3504	0.4315	0.6834	0.3293	0.5338	0.4662	0.6707
PCA = 8	0.5329	0.4988	0.4315	0.3504	0.4315	0.6834	0.3293	0.5338	0.4662	0.6707
PCA = 9	0.4671	0.5012	0.5685	0.323	0.5685	0.73	0.6707	0.4662	0.5338	0.3293
PCA = 10	0.4671	0.5012	0.5685	0.323	0.5685	0.73	0.6707	0.4662	0.5338	0.3293
PCA = 11	0.4671	0.5012	0.5685	0.323	0.5685	0.73	0.6707	0.4662	0.5338	0.3293
PCA = 12	0.5329	0.4988	0.4315	0.3504	0.4315	0.6834	0.3293	0.5338	0.4662	0.6707
PCA = 13	0.5329	0.4988	0.4315	0.3504	0.4315	0.6834	0.3293	0.5338	0.4662	0.6707
PCA = 14	0.4671	0.5012	0.5685	0.323	0.5685	0.73	0.6707	0.4662	0.5338	0.3293
PCA = 15	0.5329	0.4988	0.4315	0.3504	0.4315	0.6834	0.3293	0.5338	0.4662	0.6707
PCA = 16	0.5329	0.4988	0.4315	0.3504	0.4315	0.6834	0.3293	0.5338	0.4662	0.6707
PCA = 17	0.4671	0.5012	0.5685	0.323	0.5685	0.73	0.6707	0.4662	0.5338	0.3293
PCA = 18	0.5329	0.4988	0.4315	0.3504	0.4315	0.6834	0.3293	0.5338	0.4662	0.6707

PCA = 19	0.4671	0.5012	0.5685	0.323	0.5685	0.73	0.6707	0.4662	0.5338	0.3293
PCA = 20	0.4671	0.5012	0.5685	0.323	0.5685	0.73	0.6707	0.4662	0.5338	0.3293
PCA = 21	0.4998	0.4992	0.4975	0.3961	0.4975	0.7072	0.4946	0.5003	0.4997	0.5054
PCA = 22	0.5005	0.5008	0.5026	0.3976	0.5026	0.7068	0.5052	0.5	0.5	0.4948
PCA = 23	0.5005	0.5008	0.5026	0.3976	0.5026	0.7068	0.5052	0.5	0.5	0.4948
PCA = 24	0.5	0.4992	0.4975	0.3963	0.4975	0.7071	0.4945	0.5006	0.4994	0.5055
PCA = 25	0.4996	0.4992	0.4974	0.396	0.4974	0.7074	0.4948	0.5	0.5	0.5052
PCA = 26	0.4994	0.4992	0.4974	0.3959	0.4974	0.7075	0.4951	0.4998	0.5002	0.5049
PCA = 27	0.7399	0.4979	0.406	0.4264	0.406	0.51	0.0691	0.7428	0.2572	0.9309
PCA = 28	0.7399	0.4979	0.406	0.4264	0.406	0.51	0.0691	0.7428	0.2572	0.9309
PCA = 29	0.2601	0.5021	0.594	0.2099	0.594	0.8602	0.9309	0.2572	0.7428	0.0691

Appendix F: PCA comparison based on features for experiment 2.

dataset 2										
Models	Accuracy	Precision	Recall	F1-score	ROC auc score	RMSE	TPR	TNR	FPR	FNR
PCA = 1	0.9083	0.6025	0.8417	0.642	0.8417	0.3028	0.7708	0.9125	0.0875	0.2292
PCA = 2	0.4185	0.4814	0.3414	0.3057	0.3414	0.7626	0.2594	0.4233	0.5767	0.7406
PCA = 3	0.4133	0.4807	0.3366	0.3028	0.3366	0.766	0.255	0.4181	0.5819	0.745
PCA = 4	0.4295	0.4824	0.3496	0.3116	0.3496	0.7553	0.2647	0.4345	0.5655	0.7353
PCA = 5	0.1012	0.4096	0.1805	0.0946	0.1805	0.948	0.2647	0.0962	0.9038	0.7353
PCA = 6	0.1011	0.4123	0.1944	0.0948	0.1944	0.9481	0.2936	0.0952	0.9048	0.7064

Appendix G: PCA comparison based on features for experiment 3.

dataset 3										
Models	Accuracy	Precision	Recall	F1-score	ROC auc score	RMSE	TPR	TNR	FPR	FNR

PCA = 1	0.5329	0.4988	0.4315	0.3504	0.4315	0.6834	0.3293	0.5338	0.4662	0.6707
PCA = 2	0.5749	0.5038	0.7197	0.373	0.7197	0.652	0.8659	0.5736	0.4264	0.1341
PCA = 3	0.4004	0.4959	0.2719	0.2868	0.2719	0.7743	0.1423	0.4016	0.5984	0.8577
PCA = 4	0.4099	0.4963	0.2908	0.2917	0.2908	0.7682	0.1707	0.4109	0.5891	0.8293
PCA = 5	0.4671	0.5012	0.5685	0.323	0.5685	0.73	0.6707	0.4662	0.5338	0.3293
PCA = 6	0.53291	0.49882	0.43153	0.35038	0.43153	0.68344	0.32927	0.53379	0.46621	0.67073
PCA = 7	0.5329	0.4988	0.4315	0.3504	0.4315	0.6834	0.3293	0.5338	0.4662	0.6707
PCA = 8	0.5329	0.4988	0.4315	0.3504	0.4315	0.6834	0.3293	0.5338	0.4662	0.6707
PCA = 9	0.4671	0.5012	0.5685	0.323	0.5685	0.73	0.6707	0.4662	0.5338	0.3293
PCA = 10	0.4671	0.5012	0.5685	0.323	0.5685	0.73	0.6707	0.4662	0.5338	0.3293
PCA = 11	0.4671	0.5012	0.5685	0.323	0.5685	0.73	0.6707	0.4662	0.5338	0.3293
PCA = 12	0.5329	0.4988	0.4315	0.3504	0.4315	0.6834	0.3293	0.5338	0.4662	0.6707
PCA = 13	0.5329	0.4988	0.4315	0.3504	0.4315	0.6834	0.3293	0.5338	0.4662	0.6707

Appendix H: PCA comparison based on features for experiment 4.

dataset 4										
Models	Accuracy	Precision	Recall	F1-score	ROC auc score	RMSE	TPR	TNR	FPR	FNR
PCA = 1	0.44	0.4559	0.3778	0.3455	0.3778	0.7483	0.3	0.4556	0.5444	0.7
PCA = 2	0.56	0.5282	0.5778	0.4544	0.5778	0.6633	0.6	0.5556	0.4444	0.4
PCA = 3	0.54	0.524	0.5667	0.4415	0.5667	0.6782	0.6	0.5333	0.4667	0.4
PCA = 4	0.55	0.5101	0.5278	0.4357	0.5278	0.6708	0.5	0.5556	0.4444	0.5
PCA = 5	0.49	0.482	0.45	0.3869	0.45	0.7141	0.4	0.5	0.5	0.6
PCA = 6	0.45	0.4739	0.4278	0.3628	0.4278	0.7416	0.4	0.4556	0.5444	0.6
PCA = 7	0.45	0.4739	0.4278	0.3628	0.4278	0.7416	0.4	0.4556	0.5444	0.6
PCA = 8	0.55	0.5261	0.5722	0.4479	0.5722	0.6708	0.6	0.5444	0.4556	0.4
PCA = 9	0.55	0.52609	0.57222	0.44792	0.57222	0.67082	0.6	0.54444	0.45556	0.4

PCA = 10	0.52	0.52	0.5556	0.4286	0.5556	0.6928	0.6	0.5111	0.4889	0.4
PCA = 11	0.45	0.4739	0.4278	0.3628	0.4278	0.7416	0.4	0.4556	0.5444	0.6

Appendix I: PCA comparison based on features for experiment 6.

dataset 6										
Models	Accuracy	Precision	Recall	F1-score	ROC auc score	RMSE	TPR	TNR	FPR	FNR
PCA = 1	0.5011	0.5005	0.5134	0.3426	0.5134	0.7063	0.526	0.5009	0.4991	0.474
PCA = 2	0.5013	0.5005	0.5135	0.3427	0.5135	0.7062	0.526	0.501	0.499	0.474
PCA = 3	0.4987	0.4995	0.4865	0.3406	0.4865	0.708	0.474	0.499	0.501	0.526
PCA = 4	0.4975	0.4995	0.4859	0.3401	0.4859	0.7089	0.474	0.4977	0.5023	0.526

Appendix J: PCA comparison based on features for experiment 7.

dataset 7										
Models	Accuracy	Precision	Recall	F1-score	ROC auc score	RMSE	TPR	TNR	FPR	FNR
PCA = 1	0.5008	0.5008	0.5026	0.3978	0.5026	0.7065	0.5047	0.5005	0.4995	0.4953
PCA = 2	0.4994	0.4992	0.4974	0.3959	0.4974	0.7075	0.4951	0.4998	0.5002	0.5049
PCA = 3	0.4994	0.4992	0.4974	0.3959	0.4974	0.7075	0.4951	0.4998	0.5002	0.5049
PCA = 4	0.5003	0.4993	0.4976	0.3964	0.4976	0.7069	0.4944	0.5008	0.4992	0.5056
PCA = 5	0.4995	0.4992	0.4974	0.396	0.4974	0.7074	0.4948	0.5	0.5	0.5052
PCA = 6	0.4998	0.4992	0.4975	0.3961	0.4975	0.7072	0.4946	0.5003	0.4997	0.5054
PCA = 7	0.5005	0.5008	0.5026	0.3976	0.5026	0.7068	0.5052	0.5	0.5	0.4948
PCA = 8	0.5005	0.5008	0.5026	0.3976	0.5026	0.7068	0.5052	0.5	0.5	0.4948
PCA = 9	0.5	0.4992	0.4975	0.3963	0.4975	0.7071	0.4945	0.5006	0.4994	0.5055
PCA = 10	0.4996	0.4992	0.4974	0.396	0.4974	0.7074	0.4948	0.5	0.5	0.5052
PCA = 11	0.4996	0.4992	0.4974	0.396	0.4974	0.7074	0.4948	0.5	0.5	0.5052

PCA = 12	0.5004	0.5008	0.5026	0.3976	0.5026	0.7068	0.5052	0.5	0.5	0.4948
PCA = 13	0.5004	0.5008	0.5026	0.3976	0.5026	0.7068	0.5052	0.5	0.5	0.4948
PCA = 14	0.5004	0.5008	0.5026	0.3976	0.5026	0.7068	0.5052	0.4999	0.5001	0.4948
PCA = 15	0.5004	0.5008	0.5026	0.3976	0.5026	0.7068	0.5052	0.4999	0.5001	0.4948
PCA = 16	0.5004	0.5008	0.5026	0.3976	0.5026	0.7068	0.5052	0.5	0.5	0.4948
PCA = 17	0.5	0.5008	0.5025	0.3974	0.5025	0.7071	0.5055	0.4995	0.5005	0.4945
PCA = 18	0.4999	0.4992	0.4975	0.3962	0.4975	0.7072	0.4946	0.5004	0.4996	0.5054
PCA = 19	0.5001	0.5008	0.5025	0.3974	0.5025	0.7071	0.5055	0.4995	0.5005	0.4945
PCA = 20	0.4996	0.4992	0.4974	0.396	0.4974	0.7074	0.4948	0.5001	0.4999	0.5052
PCA = 21	0.4999	0.4992	0.4975	0.3962	0.4975	0.7072	0.4946	0.5004	0.4996	0.5054
PCA = 22	0.4999	0.4992	0.4975	0.3962	0.4975	0.7072	0.4946	0.5004	0.4996	0.5054
PCA = 23	0.4999	0.4992	0.4975	0.3962	0.4975	0.7072	0.4946	0.5004	0.4996	0.5054
PCA = 24	0.4996	0.4992	0.4974	0.396	0.4974	0.7074	0.4947	0.5001	0.4999	0.5053
PCA = 25	0.5001	0.5008	0.5025	0.3974	0.5025	0.707	0.5054	0.4996	0.5004	0.4946
PCA = 26	0.4999	0.4992	0.4975	0.3962	0.4975	0.7072	0.4946	0.5004	0.4996	0.5054
PCA = 27	0.4996	0.4992	0.4974	0.396	0.4974	0.7074	0.4947	0.5001	0.4999	0.5053
PCA = 28	0.5002	0.5008	0.5026	0.3975	0.5026	0.707	0.5054	0.4997	0.5003	0.4946
PCA = 29	0.5001	0.5008	0.5025	0.3975	0.5025	0.707	0.5054	0.4996	0.5004	0.4946
PCA = 30	0.5004	0.5008	0.5026	0.3976	0.5026	0.7068	0.5053	0.4999	0.5001	0.4947
PCA = 31	0.5004	0.5008	0.5026	0.3976	0.5026	0.7068	0.5053	0.4999	0.5001	0.4947
PCA = 32	0.5004	0.5008	0.5026	0.3976	0.5026	0.7068	0.5053	0.4999	0.5001	0.4947
PCA = 33	0.4999	0.4992	0.4975	0.3962	0.4975	0.7072	0.4946	0.5004	0.4996	0.5054
PCA = 34	0.4999	0.4992	0.4975	0.3962	0.4975	0.7072	0.4946	0.5004	0.4996	0.5054
PCA = 35	0.5004	0.5008	0.5026	0.3976	0.5026	0.7068	0.5053	0.4999	0.5001	0.4947
PCA = 36	0.4998	0.4992	0.4975	0.3962	0.4975	0.7072	0.4946	0.5003	0.4997	0.5054
PCA = 37	0.5002	0.5008	0.5025	0.3975	0.5025	0.707	0.5054	0.4997	0.5003	0.4946
PCA = 38	0.4998	0.4992	0.4975	0.3962	0.4975	0.7072	0.4946	0.5003	0.4997	0.5054

PCA = 39	0.4998	0.4992	0.4975	0.3962	0.4975	0.7072	0.4946	0.5003	0.4997	0.5054
PCA = 40	0.4996	0.4992	0.4974	0.396	0.4974	0.7074	0.4947	0.5001	0.4999	0.5053
PCA = 41	0.4996	0.4992	0.4974	0.396	0.4974	0.7074	0.4947	0.5001	0.4999	0.5053
PCA = 42	0.4996	0.4992	0.4974	0.396	0.4974	0.7074	0.4947	0.5001	0.4999	0.5053
PCA = 43	0.4996	0.4992	0.4974	0.396	0.4974	0.7074	0.4947	0.5001	0.4999	0.5053
PCA = 44	0.5004	0.5008	0.5026	0.3976	0.5026	0.7069	0.5053	0.4999	0.5001	0.4947
PCA = 45	0.5004	0.5008	0.5026	0.3976	0.5026	0.7069	0.5053	0.4999	0.5001	0.4947
PCA = 46	0.4996	0.4992	0.4974	0.396	0.4974	0.7074	0.4947	0.5001	0.4999	0.5053
PCA = 47	0.4998	0.4992	0.4975	0.3962	0.4975	0.7072	0.4946	0.5003	0.4997	0.5054
PCA = 48	0.5004	0.5008	0.5026	0.3976	0.5026	0.7069	0.5053	0.4999	0.5001	0.4947
PCA = 49	0.5004	0.5008	0.5026	0.3976	0.5026	0.7069	0.5053	0.4999	0.5001	0.4947
PCA = 50	0.4996	0.4992	0.4974	0.396	0.4974	0.7074	0.4947	0.5001	0.4999	0.5053
PCA = 51	0.4998	0.4992	0.4975	0.3961	0.4975	0.7072	0.4946	0.5003	0.4997	0.5054
PCA = 52	0.5004	0.5008	0.5026	0.3976	0.5026	0.7068	0.5053	0.4999	0.5001	0.4947
PCA = 53	0.5003	0.5008	0.5026	0.3976	0.5026	0.7069	0.5053	0.4999	0.5001	0.4947
PCA = 54	0.5003	0.5008	0.5026	0.3976	0.5026	0.7069	0.5053	0.4999	0.5001	0.4947
PCA = 55	0.5004	0.5008	0.5026	0.3976	0.5026	0.7069	0.5053	0.4999	0.5001	0.4947
PCA = 56	0.4998	0.4992	0.4975	0.3961	0.4975	0.7072	0.4946	0.5003	0.4997	0.5054
PCA = 57	0.5002	0.5008	0.5025	0.3975	0.5025	0.707	0.5054	0.4997	0.5003	0.4946
PCA = 58	0.5002	0.5008	0.5025	0.3975	0.5025	0.707	0.5054	0.4997	0.5003	0.4946

Appendix K: PCA comparison based on features for experiment 8.

dataset 8										
Models	Accuracy	Precision	Recall	F1-score	ROC auc score	RMSE	TPR	TNR	FPR	FNR
PCA = 25	0.4904	0.4861	0.4796	0.4438	0.4796	0.7139	0.4604	0.4987	0.5013	0.5396
PCA = 50	0.491	0.4891	0.484	0.4459	0.484	0.7135	0.4716	0.4964	0.5036	0.5284

PCA = 75	0.4989	0.499	0.4985	0.4558	0.4985	0.7079	0.4978	0.4992	0.5008	0.5022
PCA = 100	0.5062	0.5149	0.5218	0.4685	0.5218	0.7027	0.5495	0.4941	0.5059	0.4505
PCA = 125	0.4943	0.4918	0.4881	0.4492	0.4881	0.7112	0.4771	0.4991	0.5009	0.5229
PCA = 150	0.494	0.4878	0.4822	0.4466	0.4822	0.7113	0.4612	0.5031	0.4969	0.5388
PCA = 175	0.503	0.5036	0.5053	0.4605	0.5053	0.705	0.5092	0.5013	0.4987	0.4908
PCA = 200	0.494	0.49	0.4853	0.4479	0.4853	0.7113	0.4699	0.5007	0.4993	0.5301

Appendix L: PCA comparison based on features for experiment 9.

dataset 9										
Models	Accuracy	Precision	Recall	F1-score	ROC auc score	RMSE	TPR	TNR	FPR	FNR
PCA = 25	0.5171	0.5117	0.5115	0.5108	0.5115	0.6949	0.4366	0.5864	0.4136	0.5634
PCA = 50	0.517	0.5116	0.5114	0.5108	0.5114	0.695	0.4366	0.5862	0.4138	0.5634
PCA = 75	0.517	0.5115	0.5113	0.5107	0.5113	0.695	0.4364	0.5862	0.4138	0.5636
PCA = 100	0.4831	0.4886	0.4888	0.4824	0.4888	0.7189	0.5639	0.4136	0.5864	0.4361
PCA = 125	0.5169	0.5114	0.5112	0.5105	0.5112	0.6951	0.4361	0.5864	0.4136	0.5639

Appendix M: All results in K-means Model for Experiment 1.

creditcard dataset	
two clusters method	
tunimg parameters	**evaluations**

initialization	n_init	max iter	algorithm	random state	accuracy	precision	recall	f1-score	roc auc score	rmse	saa	tpr	tnr	fpr	fnr
K-means++	5	1	auto	0	0.5026	0.498	0.3839	0.3365	0.3839	0.7052	50.264	0.2642	0.5037	0.4963	0.7358
K-means++	5	10	auto	0	0.5256	0.4983	0.4036	0.3468	0.4036	0.6888	52.557	0.2805	0.5266	0.4734	0.7195
K-means++	5	1	auto	42	0.4637	0.503	0.672	0.3228	0.672	0.7323	46.37	0.882114	0.461917	0.5381	0.1179
K-means++	5	10	auto	42	0.4744	0.5017	0.5964	0.3268	0.5964	0.725	47.443	0.7195	0.4734	0.5266	0.2805
K-means++	5	1	auto	1	0.5512	0.5002	0.5136	0.3595	0.5136	0.6699	55.117	0.4756	0.5515	0.4485	0.5244
K-means++	5	10	auto	1	0.4744	0.5017	0.5964	0.3268	0.5964	0.725	47.443	0.7195	0.4734	0.5266	0.2805
K-means++	5	1	auto	2	0.2495	0.4974	0.3884	0.2016	0.3884	0.8663	24.947	0.5285	0.2483	0.7517	0.4715
K-means++	5	10	auto	2	0.5256	0.4983	0.4036	0.3468	0.4036	0.6888	52.557	0.2805	0.5266	0.4734	0.7195
K-means++	5	1	auto	3	0.5361	0.5002	0.5141	0.3531	0.5141	0.6811	53.608	0.4919	0.5363	0.4637	0.5081
K-means++	5	10	auto	3	0.5256	0.4983	0.4036	0.3468	0.4036	0.6888	52.557	0.2805	0.5266	0.4734	0.7195
K-means++	5	1	auto	4	0.5552	0.5003	0.5196	0.3613	0.5196	0.6669	55.524	0.4837	0.5555	0.4445	0.5163
K-means++	5	10	auto	4	0.6102	0.5002	0.5088	0.3831	0.5088	0.6243	61.02	0.4065	0.6111	0.3889	0.5935
K-means++	5	1	auto	5	0.6623	0.5032	0.6685	0.4065	0.6685	0.5811	66.229	0.674797	0.662241	0.3378	0.3252

K-means++	5	10	auto	5	0.5256	0.4983	0.4036	0.3468	0.4036	0.6888	52.557	0.2805	0.5266	0.4734	0.7195
K-means++	5	1	auto	13	0.779	0.5059	0.7372	0.4509	0.7372	0.4701	77.9	0.695122	0.779364	0.2206	0.3049
K-means++	5	10	auto	13	0.4744	0.5017	0.5964	0.3268	0.5964	0.725	47.443	0.7195	0.4734	0.5266	0.2805
K-means++	5	1	auto	14	0.9905	0.4978	0.4974	0.4976	0.4974	0.0975	99.049	0	0.994777	0.0052	1
K-means++	5	10	auto	14	0.5254	0.4984	0.4075	0.3468	0.4075	0.6889	52.542	0.2886	0.5264	0.4736	0.7114
K-means++	5	1	auto	90	0.0118	0.4786	0.4592	0.0118	0.4592	0.9941	1.1767	0.910569	0.007879	0.9921	0.0894
K-means++	5	10	auto	90	0.5256	0.4983	0.4036	0.3468	0.4036	0.6888	52.556	0.2805	0.5266	0.4734	0.7195
K-means++	5	1	auto	91	0.9829	0.5097	0.53	0.5135	0.53	0.1307	98.291	0.073171	0.986846	0.0132	0.9268
K-means++	5	10	auto	91	0.4744	0.5017	0.5964	0.3268	0.5964	0.725	47.443	0.7195	0.4734	0.5266	0.2805
K-means++	5	1	auto	200	0.3517	0.5027	0.6441	0.2649	0.6441	0.8052	35.17	0.939024	0.349155	0.6508	0.061
K-means++	5	10	auto	200	0.4744	0.5017	0.5964	0.3268	0.5964	0.725	47.443	0.7195	0.4734	0.5266	0.2805
K-means++	5	1	auto	250	0.951	0.5103	0.6071	0.5093	0.6071	0.2214	95.097	0.260163	0.9539	0.046	0.7398
K-means++	5	10	auto	250	0.474443	0.501663	0.596443	0.326853	0.596443	0.724963	47.4426	0.719511	0.473377	0.526633	0.280493
random	5	1	auto	5	0.3249	0.5029	0.6468	0.2497	0.6468	0.8217	32.485	0.9715	0.3221	0.6779	0.0285
random	5	10	full	5	0.4744	0.5017	0.5964	0.3268	0.5964	0.725	47.443	0.7195	0.4734	0.5266	0.2805

random	5	10	elkan	5	0.2495	0.4974	0.3884	0.2016	0.3884	0.8663	24.947	0.5285	0.2483	0.7517	0.4715
random	10	1	auto	5	0.5256	0.4983	0.4036	0.3468	0.4036	0.6888	52.557	0.2805	0.5266	0.4734	0.7195
random	5	10	auto	5	0.5361	0.5002	0.5141	0.3531	0.5141	0.6811	53.608	0.4919	0.5363	0.4637	0.5081

Appendix N: All results in HMM Model for Experiment 1.

creditcard dataset														
two states method														
Tuning Parameters					**Evaluations**									
covariance_type	**min_covar**	**n_iter**	**algorithm**	**tol**	**Accuracy**	**Precision**	**Recall**	**F1-score**	**ROC auc score**	**RMSE**	**TPR**	**TNR**	**FPR**	**FNR**
spherical	0.0001	5000	viterbi	0.1	0.92689	0.52471	0.9066	0.5282	0.906624594	0.27038	0.88618	0.92707	0.073	0.11382
diag	0.0001	5000	viterbi	0.1	0.8432	0.512	0.877	0.481	0.8767537	0.3959	0.91057	0.842938	0.16	0.0894
tied	0.0001	5000	viterbi	0.1	0.5166	0.4978	0.375	0.342	0.3747566	0.6953	0.232	0.5178	0.48	0.7683
full	0.0001	defults	viterbi	defults	0.6836	0.5056	0.786	0.417	0.7864687	0.5625	0.89024	0.682693	0.32	0.1098
spherical	0.0001	defults	viterbi	defults	0.8963	0.5178	0.897	0.507	0.8973431	0.322	0.89837	0.896312	0.1	0.1016
diag	0.0001	defults	viterbi	defults	0.1569	0.488	0.123	0.136	0.1232903	0.9182	0.08943	0.1571	0.84	0.9106
tied	0.0001	defults	viterbi	defults	0.4834	0.5022	0.625	0.331	0.6252434	0.7187	0.76829	0.4822	0.52	0.2317
spherical	0.0001	5000	map	0.1	0.8963	0.5178	0.897	0.507	0.8973431	0.322	0.898	0.8963	0.1	0.1016

diag	0.0001	5000	map	0.1	0.1569	0.488	0.123	0.136	0.1232903	0.9182	0.089	0.1571	0.84	0.9106
tied	0.0001	5000	map	0.1	0.5166	0.4978	0.375	0.342	0.3747654	0.6953	0.232	0.5178	0.48	0.7683
full	0.0001	defults	map	defults	0.3164	0.4944	0.214	0.241	0.2135313	0.8268	0.11	0.3173	0.68	0.8902
spherical	0.0001	defults	map	defults	0.1037	0.4822	0.103	0.094	0.1026569	0.9467	0.102	0.1037	0.9	0.8984
diag	0.0001	defults	map	defults	0.48341	0.50215	0.6252	0.33 14	0.625234571	0.71874	0.7683	0.48218	0.518	0.23171
tied	0.0001	defults	map	defults	0.5166	0.4978	0.375	0.342	0.3747654	0.6953	0.232	0.5178	0.48	0.7683
spherical	0.0001	5000	viterbi	defults	0.07311	0.47529	0.0934	0.0683	0.093375406	0.96275	0.1138	0.07293	0.927	0.88618
spherical	0.0001	5	viterbi	0.1	0.2175	0.4912	0.146	0.179	0.1456282	0.8846	0.073	0.2181	0.78	0.9268

Appendix O: All results in Auto-Encoder Model for Experiment 1.

Threshold Method																		
Tuning Parameters									**Evaluations**									
nb_epoch	**batch_size**	**input_dim**	**encoding_dim**	**hidden_dim1**	**hidden_dim2**	**activation**	**learning_rate**	**Threshold**	**Accuracy**	**Precision**	**Recall**	**F1-score**	**ROC auc score**	**RMSE**	**TPR**	**TNR**	**FPR**	**FNR**

10	128	30	18	10	6	tanh	1.00E-07	4	0.9813	0.5769	0.8671	0.6237	0.8671	0.1369	0.752033	0.982256	0.0177	0.248
50	128	30	18	10	6	tanh	1.00E-07	4	0.9822	0.5765	0.8413	0.622	0.8413	0.1334	0.6991887	0.983434	0.0166	0.3008
10	128	30	32	16	8	tanh	1.00E-07	4	0.9825	0.5774	0.8394	0.6231	0.8394	0.1323	0.6951122	0.98375	0.0162	0.3049
10	128	30	10	5	2	tanh	1.00E-07	4	0.9799	0.5742	0.8806	0.6203	0.8806	0.1417	0.7804888	0.980778	0.0192	0.2195
10	128	30	5	2	1	tanh	1.00E-07	4	0.9779	0.5696	0.8918	0.6137	0.8918	0.1487	0.8048788	0.978633	0.0214	0.1951
10	128	30	5	3	1	tanh	1.00E-07	4	0.9781	0.5669	0.8656	0.6088	0.8656	0.1478	0.752033	0.979125	0.0209	0.248
10	128	30	50	20	10	tanh	1.00E-07	4	0.9833	0.5804	0.8378	0.6272	0.8378	0.1292	0.6910557	0.984559	0.0154	0.3089
10	12	30	50	20	10	tanh	1.00E-07	4	0.9839	0.5836	0.8401	0.6315	0.8401	0.1269	0.6951122	0.985157	0.0148	0.3049
10	12	30	5	2	1	tanh	1.00E-07	4	0.9791	0.572	0.8842	0.6172	0.8842	0.1447	0.7886188	0.979881	0.0201	0.2114
10	256	30	5	2	1	tanh	1.00E-07	4	0.9774	0.5665	0.8774	0.6085	0.8774	0.1503	0.7764233	0.978281	0.0217	0.2236

10	128	30	5	2	1	sigmoid	1.00E-07	4	0.9757	0.5616	0.8724	0.6007	0.8724	0.156	0.7682933	0.9765588	0.02344	0.23177
10	128	30	5	2	1	hard_sigmoid	1.00E-07	4	0.9757	0.5612	0.8684	0.6	0.8684	0.1558	0.7601633	0.9766466	0.02344	0.23988
10	128	30	5	2	1	exponential	1.00E-07	4	0.9757	0.5612	0.8684	0.6	0.8684	0.1558	0.7601633	0.9766466	0.02344	0.23988
10	128	30	5	2	1	linear	1.00E-07	4	0.9795	0.5713	0.8683	0.61556	0.8683	0.143	0.7560998	0.9805115	0.0195	0.24339
10	128	30	5	2	1	tanh	1.00E-07	3	0.9713	0.5561	0.8986	0.59211	0.8986	0.1693	0.8252033	0.9719688	0.028	0.17488
10	128	30	5	2	1	tanh	1.00E-07	2	0.9536	0.5378	0.916	0.55822	0.916	0.2153	0.8780449	0.95396	0.046	0.122
10	128	30	5	2	1	tanh	1.00E-07	1	0.8365	0.5117	0.8794	0.47855	0.8794	0.4044	0.9227764	0.83608	0.1639	0.07722
10	128	30	5	2	1	tanh	1.00E-07	5	0.9825	0.5741	0.8192	0.61744	0.8192	0.1323	0.6544472	0.9839099	0.0161	0.34555
10	128	30	5	2	1	linear	1.00E-06	4	0.9802	0.5734	0.8686	0.61887	0.8686	0.1407	0.7560998	0.9811165	0.0188	0.24339
10	128	30	5	2	1	tanh	1.00E-08	4	0.9817	0.5707	0.8168	0.61255	0.8168	0.1353	0.6510407	0.9831117	0.0169	0.34996

10	128	30	5	2	1	tanh	1.00E-09	4	0.9817	0.5715	0.8208	0.6139	0.8208	0.1351	0.6585377	0.9831355	0.01699	0.34155
10	128	30	5	2	1	tanh	1.00E-06	4	0.9819	0.5729	0.825	0.6616	0.825	0.1344	0.6666667	0.9832933	0.0167	0.33333

Appendix P: All results in K-means Model for Experiment 2.

two clusters method													
Tuning Parameters					**Evaluations**								
initialization	**n_init**	**max_iter**	**algorithm**	**RandomState**	**Accuracy**	**Precision**	**Recall**	**F1-score**	**ROC auc score**	**RMSE**	**TPR**	**TNR**	**FPR**	**FNR**
K-means++	5	1	auto	0	0.3243	0.4699	0.2649	0.2513	0.2649	0.822	0.2017	0.3281	0.6719	0.7983
K-means++	5	10	auto	0	0.4115	0.4804	0.334	0.3017	0.334	0.7672	0.2517	0.4164	0.5836	0.7483
K-means++	5	1	auto	42	0.317	0.4947	0.4611	0.2587	0.4611	0.8264	0.6142	0.3079	0.6921	0.3858
K-means++	5	10	auto	42	0.4137	0.4808	0.3372	0.303	0.3372	0.7657	0.2558	0.4186	0.5814	0.7442
K-means++	5	1	auto	1	0.4266	0.4882	0.3993	0.3144	0.3993	0.7572	0.3703	0.4283	0.5717	0.6297
K-means++	5	10	auto	1	0.4163	0.4811	0.3397	0.3045	0.3397	0.764	0.2583	0.4212	0.5788	0.7417
K-means++	5	1	auto	2	0.5707	0.5136	0.6161	0.4019	0.6161	0.6552	0.6644	0.5678	0.4322	0.3356

K-means++	5	10	auto	2	0.5833	0.5188	0.6599	0.4125	0.6599	0.6455	0.7414	0.5785	0.4215	0.2586
K-means++	5	1	auto	3	0.9007	0.5993	0.8576	0.6368	0.8576	0.3151	0.8117	0.9035	0.0965	0.1883
K-means++	5	10	auto	3	0.5794	0.5184	0.657	0.4104	0.657	0.6485	0.7394	0.5745	0.4255	0.2606
K-means++	5	1	auto	4	0.5639	0.5125	0.6067	0.3979	0.6067	0.6603	0.6522	0.5612	0.4388	0.3478
K-means++	5	10	auto	4	0.5838	0.5189	0.6605	0.4128	0.6605	0.6451	0.7419	0.579	0.421	0.2581
K-means++	5	1	auto	5	0.6884	0.5306	0.7343	0.4701	0.7343	0.5583	0.7831	0.6855	0.3145	0.2169
K-means++	5	10	auto	5	0.5888	0.5196	0.6665	0.4157	0.6665	0.6413	0.7492	0.5839	0.4161	0.2508
K-means++	5	1	auto	13	0.0987	0.4004	0.1427	0.0918	0.1427	0.9494	0.1894	0.0959	0.9041	0.8106
K-means++	5	10	auto	13	0.4195	0.4815	0.3423	0.3063	0.3423	0.7619	0.2603	0.4244	0.5756	0.7397
K-means++	5	1	auto	14	0.5089	0.5126	0.6095	0.3725	0.6095	0.7008	0.7164	0.5026	0.4974	0.2836
K-means++	5	10	auto	14	0.5811	0.5186	0.6585	0.4113	0.6585	0.6472	0.7408	0.5762	0.4238	0.2592
K-means++	5	1	auto	90	0.4473	0.4919	0.4302	0.327	0.4302	0.7434	0.4119	0.4484	0.5516	0.5881
K-means++	5	10	auto	90	0.4185	0.4814	0.3414	0.3057	0.3414	0.7626	0.2594	0.4234	0.5766	0.7406
K-means++	5	1	auto	91	0.4871	0.4865	0.3828	0.3409	0.3828	0.7162	0.2719	0.4937	0.5063	0.7281
K-means++	5	10	auto	91	0.4184	0.4814	0.3415	0.3056	0.3415	0.7626	0.2597	0.4232	0.5768	0.7403

K-means++	5	1	auto	200	0.9012	0.5995	0.8566	0.6371	0.8566	0.3143	0.8092	0.9041	0.0959	0.1908
K-means++	5	10	auto	200	0.4116	0.4804	0.3341	0.3017	0.3341	0.7671	0.2517	0.4165	0.5835	0.7483
K-means++	5	1	auto	250	0.3467	0.4845	0.3775	0.271	0.3775	0.8083	0.4103	0.3447	0.6553	0.5897
K-means++	5	10	auto	250	0.4133	0.4807	0.3366	0.3028	0.3366	0.766	0.255	0.4181	0.5819	0.74
random	5	1	Full	5	0.4195	0.4815	0.3423	0.3063	0.3423	0.7619	0.2603	0.4244	0.5756	0.7397
random	5	10	elkan	5	0.5089	0.5126	0.6095	0.3725	0.6095	0.7008	0.7164	0.5026	0.4974	0.2836
random	10	10	auto	5	0.5811	0.5186	0.6585	0.4113	0.6585	0.6472	0.7408	0.5762	0.4238	0.2592

Appendix Q: All results in HMM Model for Experiment 2.

two states method														
Tuning Parameters						**Evaluations**								
covariance_type	**min_covar**	**n_iter**	**algorithm**	**tol**	**Random State**	**Accuracy**	**Precision**	**Recall**	**F1-score**	**ROC auc score**	**RMSE**	**TPR**	**TNR**	**FPR**	**FNR**
spherical	0.0001	5000	viterbi	0.1	defaults	0.8885	0.5878	0.842	0.618	0.8417246	0.3338	0.792	0.8915	0.11	0.2081
diag	0.0001	5000	viterbi	0.1	defaults	0.8632	0.5893	0.93	0.614	0.9295126	0.3698	1	0.859	0.14	0
tied	0.0001	5000	viterbi	0.1	defaults	0.115	0.427	0.224	0.107	0.2240561	0.9407	0.34	0.1081	0.89	0.6
full	0.0001	5000	viterbi	defaults	defaults	0.8632	0.5893	0.93	0.614	0.9295126	0.3698	1	0.859	0.14	0

spherical	0.0001		viterbi	defaults	defaults	0.8885	0.5878	0.842	0.618	0.8417246	0.3338	0.792	0.8915	0.11	0.2081
diag	0.0001		viterbi	defaults	defaults	0.8632	0.5893	0.93	0.614	0.9295126	0.3698	1	0.859	0.14	0
full	0.0001		viterbi	defaults	defaults	0.1368	0.4107	0.07	0.12	0.0704874	0.9291	0	0.141	0.86	1
tied	0.0001		viterbi	defaults	defaults	0.885	0.573	0.776	0.596	0.7759439	0.3391	0.66	0.8919	0.11	0.3
spherical	0.0001	5000	map	0.1	defaults	0.8885	0.5878	0.842	0.618	0.8417246	0.3338	0.792	0.8915	0.11	0.2081
diag	0.0001	5000	map	0.1	defaults	0.1368	0.4107	0.07	0.12	0.0704874	0.9291	0	0.141	0.86	1
tied	0.0001	5000	map	0.1	defaults	0.115	0.427	0.224	0.107	0.2240561	0.9407	0.34	0.1081	0.89	0.6
full	0.0001	5000	map	0.1	defaults	0.5399	0.491	0.423	0.367	0.4229737	0.6783	0.299	0.5473	0.45	0.7014
spherical	0.0001		map	defults	defaults	0.1115	0.4122	0.158	0.103	0.1582754	0.9426	0.208	0.1085	0.89	0.7919
diag	0.0001		map	defults	defaults	0.4597	0.5083	0.571	0.345	0.5714707	0.735	0.69	0.4527	0.55	0.3097
tied	0.0001		map	defults	defaults	0.115	0.427	0.224	0.107	0.2240561	0.9407	0.34	0.1081	0.89	
full	0.0001		map	defults	defaults	0.1368	0.4107	0.07	0.12	0.0704874	0.9291	0	0.141	0.86	1
spherical	0.0001	5000	viterbi	defults	defaults	0.1115	0.4122	0.158	0.103	0.1582754	0.9426	0.208	0.1085	0.89	0.7919
spherical	0.0001	5	viterbi	0.1	defaults	0.8885	0.5878	0.842	0.618	0.8415857	0.3339	0.792	0.8915	0.11	0.2083
spherical	0.0001	5	viterbi	0.1	42	0.5385	0.4882	0.399	0.364	0.3985421	0.6793	0.25	0.5474	0.45	0.7503

Appendix R: All results in Auto-Encoder Model for Experiment 2.

Threshold Method																		
Tuning Parameters									**Evaluations**									
nb_epoch	**batch_size**	**Input_dim**	**Encoding_dim**	**Hidden_dim1**	**Hidden_dim2**	**activation**	**Learning_rate**	**Threshold**	**Accuracy**	**Precision**	**Recall**	**F1-score**	**ROC auc score**	**RMSE**	**TPR**	**TNR**	**FPR**	**FNR**
10	128	6	18	10	6	tanh	1.00 E-07	5	0.9559	0.6327	0.6491	0.6403	0.6491	0.2099	0.3228	0.9753	0.0247	0.6772
10	128	6	32	16	8	tanh	1.00 E-07	5	0.9559	0.6327	0.6491	0.6403	0.6491	0.2099	0.3228	0.9753	0.0247	0.6772
10	128	6	10	5	2	tanh	1.00 E-07	5	0.9559	0.6327	0.6491	0.6403	0.6491	0.2099	0.3228	0.9753	0.0247	0.6772
10	128	6	5	2	1	tanh	1.00 E-07	5	0.9559	0.6327	0.6491	0.6403	0.6491	0.2099	0.3228	0.9753	0.0247	0.6772
10	128	6	5	3	1	tanh	1.00 E-07	5	0.9559	0.6327	0.6491	0.6403	0.6491	0.2099	0.3228	0.9753	0.0247	0.6772
10	128	6	50	20	10	tanh	1.00 E-07	5	0.9559	0.6327	0.6491	0.6403	0.6491	0.2099	0.3228	0.9753	0.0247	0.6772
10	128	6	5	2	1	sigmoid	1.00 E-07	5	0.9555	0.6333	0.6547	0.6431	0.6547	0.211	0.335	0.9745	0.0255	0.66
10	128	6	5	2	1	hard_sigmoid	1.00 E-07	5	0.9555	0.6333	0.6547	0.6431	0.6547	0.211	0.335	0.9745	0.0255	0.66
10	128	6	5	2	1	exponential	1.00 E-07	5	0.9555	0.6333	0.6547	0.6431	0.6547	0.211	0.335	0.9745	0.0255	0.66
10	128	6	5	2	1	linear	1.00 E-07	5	0.9703	0.4851	0.5	0.4925	0.5	0.1724	0	1	0	1
10	128	6	5	2	1	tanh	1.00 E-07	3	0.931	0.5965	0.7108	0.6275	0.7108	0.2626	0.4767	0.945	0.055	0.5233

10	128	6	5	2	1	tanh	1.00 E-07	2	0.926 6	0.603 5	0.7 589	0.6 405	0.7 589	0.2 71	0.5 806	0.9 372	0.0 628	0.4 194
10	128	6	5	2	1	tanh	1.00 E-07	1	0.921 3	0.609 8	0.8 134	0.6 518	0.8 134	0.2 806	0.6 986	0.9 281	0.0 719	0.3 014
10	128	6	5	2	1	tanh	1.00 E-07	4	0.942 6	0.603 9	0.6 73	0.6 281	0.6 73	0.2 395	0.3 864	0.9 597	0.0 403	0.6 136
10	128	6	5	2	1	linear	1.00 E-08	4	0.970 3	0.485 1	0.5	0.4 925	0.5	0.1 724	0	1	0	1
10	128	6	5	2	1	linear	1.00 E-06	4	0.970 3	0.485 1	0.5	0.4 925	0.5	0.1 724	0	1	0	1
10	128	6	5	2	1	tanh	1.00 E-08	4	0.942 6	0.603 9	0.6 73	0.6 281	0.6 73	0.2 395	0.3 864	0.9 597	0.0 403	0.6 136
10	128	6	5	2	1	tanh	1.00 E-09	4	0.942 6	0.603 9	0.6 73	0.6 281	0.6 73	0.2 395	0.3 864	0.9 597	0.0 403	0.6 136
10	128	6	5	2	1	tanh	1.00 E-06	4	0.942 6	0.603 9	0.6 73	0.6 281	0.6 73	0.2 395	0.3 864	0.9 597	0.0 403	0.6 136
50	128	6	5	2	1	tanh	1.00 E-07	4	0.942 6	0.603 9	0.6 73	0.6 281	0.6 73	0.2 395	0.3 864	0.9 597	0.0 403	0.6 136
10	256	6	5	2	1	tanh	1.00 E-07	4	0.942 6	0.603 9	0.6 73	0.6 281	0.6 73	0.2 395	0.3 864	0.9 597	0.0 403	0.6 136

Appendix S: All results in K-means Model for Experiment 3.

two clusters method														
Tuning Parameters					**Evaluations**									
initialization	**n_init**	**max_iter**	**algorithm**	**RandomState**	**Accuracy**	**Precision**	**Recall**	**F1-score**	**ROC auc score**	**RMSE**	**TPR**	**TNR**	**FPR**	**FNR**
K-means++	5	1	auto	0	0.3	0.4599	0.438 1	0.290 7	0.438 1	0.83 67	0.24 29	0.633 3	0.366 7	0.75 71

K-means++	5	10	auto	0	0.3024	0.4651	0.4464	0.2935	0.4464	0.8352	0.2429	0.65	0.35	0.7571
K-means++	5	1	auto	42	0.7317	0.5205	0.5252	0.5208	0.5252	0.518	0.8171	0.2333	0.7667	0.1829
K-means++	5	10	auto	42	0.6976	0.5349	0.5536	0.5317	0.5536	0.5499	0.7571	0.35	0.65	0.2429
K-means++	5	1	auto	1	0.739	0.5137	0.5157	0.514	0.5157	0.5109	0.8314	0.2	0.8	0.1686
K-means++	5	10	auto	1	0.2927	0.4563	0.4338	0.2844	0.4338	0.841	0.2343	0.6333	0.3667	0.7657
K-means++	5	1	auto	2	0.6927	0.5327	0.5507	0.5284	0.5507	0.5544	0.7514	0.35	0.65	0.2486
K-means++	5	10	auto	2	0.3	0.4599	0.4381	0.2907	0.4381	0.8367	0.2429	0.6333	0.3667	0.7571
K-means++	5	1	auto	3	0.2951	0.4616	0.4421	0.2873	0.4421	0.8396	0.2343	0.65	0.35	0.7657
K-means++	5	10	auto	3	0.2927	0.4563	0.4338	0.2844	0.4338	0.841	0.2343	0.6333	0.3667	0.7657
K-means++	5	1	auto	4	0.1756	0.4718	0.4895	0.1677	0.4895	0.908	0.0457	0.9333	0.0667	0.9543
K-means++	5	10	auto	4	0.7	0.5401	0.5619	0.5376	0.5619	0.5477	0.7571	0.3667	0.6333	0.2429
K-means++	5	1	auto	5	0.2561	0.4724	0.4676	0.2553	0.4676	0.8625	0.1686	0.7667	0.2333	0.8314
K-means++	5	10	auto	5	0.3	0.4599	0.4381	0.2907	0.4381	0.8367	0.2429	0.6333	0.3667	0.7571
K-means++	5	1	auto	13	0.7854	0.5351	0.529	0.5308	0.529	0.4633	0.8914	0.1667	0.8333	0.1086
K-means++	5	10	auto	13	0.3024	0.4651	0.4464	0.2935	0.4464	0.8352	0.2429	0.65	0.35	0.7571

K-means++	5	1	auto	14	0.7171	0.5362	0.5512	0.5363	0.5512	0.5319	0.7857	0.3167	0.6833	0.2143
K-means++	5	10	auto	14	0.7073	0.5437	0.5662	0.5427	0.5662	0.541	0.7657	0.3667	0.6333	0.2343
K-means++	5	1	auto	90	0.7073	0.5437	0.5662	0.5427	0.5662	0.541	0.7657	0.3667	0.6333	0.2343
K-means++	5	10	auto	90	0.7024	0.5289	0.5426	0.5263	0.5426	0.5455	0.7686	0.3167	0.6833	0.2314
K-means++	5	1	auto	91	0.71463	0.5392	0.55667	0.53915	0.55667	0.5342	0.78	0.33333	0.66667	0.22
K-means++	5	10	auto	91	0.3	0.4599	0.4381	0.2907	0.4381	0.8367	0.2429	0.6333	0.3667	0.7571
K-means++	5	1	auto	92	0.6951	0.4978	0.4969	0.4927	0.4969	0.5522	0.7771	0.2167	0.7833	0.2229
K-means++	5	10	auto	92	0.7	0.5401	0.5619	0.5376	0.5619	0.5477	0.7571	0.3667	0.6333	0.2429
K-means++	5	1	auto	200	0.6902	0.5275	0.5424	0.5225	0.5424	0.5566	0.7514	0.3333	0.6667	0.2486
K-means++	5	10	auto	200	0.3	0.4599	0.4381	0.2907	0.4381	0.8367	0.2429	0.6333	0.3667	0.7571
K-means++	5	1	auto	250	0.2634	0.4877	0.4857	0.2628	0.4857	0.8582	0.1714	0.8	0.2	0.8286
K-means++	5	10	auto	250	0.7073	0.5437	0.5662	0.5427	0.5662	0.541	0.7657	0.3667	0.6333	0.2343
random	5	1	Full	5	0.7073	0.5437	0.5662	0.5427	0.5662	0.541	0.7657	0.3667	0.6333	0.2343
random	5	10	elkan	5	0.7024	0.5289	0.5426	0.5263	0.5426	0.5455	0.7686	0.3167	0.6833	0.2314
random	10	10	auto	5	0.71463	0.5392	0.55667	0.53915	0.55667	0.5342	0.78	0.33333	0.66667	0.22

Appendix T: All results in HMM Model for Experiment 3.

two states method															
Tuning Parameters						**Evaluations**									
covariance_type	**min_covar**	**n_iter**	**algorithm**	**tol**	**Random State**	**Accuracy**	**Precision**	**Recall**	**F1-score**	**ROC auc score**	**RMSE**	**TPR**	**TNR**	**FPR**	**FNR**
spherical	0.0001	5000	viterbi	0.1	defaults	0.7488	0.5307	0.535	0.532	0.5352381	0.5012	0.837	0.2333	0.77	0.1629
diag	0.0001	5000	viterbi	0.1	defaults	0.2951	0.4834	0.477	0.291	0.4766667	0.8396	0.22	0.7333	0.27	0.78
tied	0.0001	5000	viterbi	0.1	defaults	0.7049	0.53	0.544	0.528	0.5440476	0.5433	0.771	0.3167	0.68	0.2286
full	0.0001	defaults	viterbi	0.1	defaults	0.3756	0.4754	0.455	0.347	0.4547619	0.7902	0.343	0.5667	0.43	0.6571
spherical	0.0001	defaults	viterbi	defaults	defaults	0.539	0.5046	0.509	0.45	0.5090476	0.679	0.551	0.4667	0.53	0.4486
diag	0.0001	defaults	viterbi	defaults	defaults	0.6512	0.4993	0.499	0.485	0.4988095	0.5906	0.714	0.2833	0.72	0.2857
tied	0.0001	defaults	viterbi	defaults	defaults	0.7049	0.5343	0.551	0.532	0.5509524	0.5433	0.769	0.3333	0.67	0.2314
full	0.0001	defaults	viterbi	defaults	defaults	0.622	0.5238	0.544	0.499	0.5438095	0.6149	0.654	0.4333	0.57	0.3457
spherical	0.0001	5000	map	0.1	defaults	0.7561	0.5357	0.54	0.537	0.5395238	0.4939	0.846	0.2333	0.77	0.1543
diag	0.0001	5000	map	0.1	defaults	0.7122	0.5201	0.528	0.519	0.527619	0.5365	0.789	0.2667	0.73	0.2114
tied	0.0001	5000	map	0.1	defaults	0.7049	0.53	0.544	0.528	0.5440476	0.5433	0.771	0.3167	0.68	0.2286
full	0.0001	defaults	map	0.1	defaults	0.6244	0.5246	0.545	0.501	0.5452381	0.6129	0.657	0.4333	0.57	0.3429

spherical	0.0001	defaults	map	defaults	defaults	0.539	0.5046	0.509	0.45	0.5090476	0.679	0.551	0.4667	0.53	0.4486
diag	0.0001	defaults	map	defaults	defaults	0.6415	0.4959	0.493	0.479	0.4930952	0.5988	0.703	0.2833	0.72	0.2971
tied	0.0001	defaults	map	defaults	defaults	0.2951	0.4657	0.449	0.288	0.4490476	0.8396	0.231	0.6667	0.33	0.7686
full	0.0001	defaults	map	defaults	defaults	0.378	0.4762	0.456	0.349	0.4561905	0.7886	0.346	0.5667	0.43	0.6543
spherical	0.0001	5000	viterbi	defaults	defaults	0.7634	0.5355	0.537	0.536	0.5369048	0.4864	0.857	0.2167	0.78	0.1429
spherical	0.0001	5	viterbi	0.1	defaults	0.5244	0.5072	0.514	0.445	0.5142857	0.6896	0.529	0.5	0.5	0.4714
spherical	0.0001	5	viterbi	0.1	42	0.4756	0.4928	0.486	0.412	0.4857143	0.7241	0.471	0.5	0.5	0.5286

Appendix U: All results in Auto-Encoder Model for Experiment 3.

Threshold Method																	
Tuning Parameters									**Evaluations**								
nb_epoch	**batch_size**	**input_dim**	**encoding_dim**	**hidden_dim1**	**hidden_dim2**	**activation**	**learning_rate**	**Threshold**	**Accuracy**	**Precision**	**Recall**	**F1-score**	**ROC auc score**	**RMSE**	**TPR**	**TNR**	**FPR**	**FNR**
10	128	24	18	10	6	tanh	1.00E-07	4	0.1878	0.4927	0.4967	0.1818	0.4967	0.9012	0.06	0.9333	0.0667	0.94

50	128	24	18	10	6	tanh	1.00E-07	4	0.1854	0.5475	0.516	0.1767	0.516	0.9026	0.0486	0.9833	0.0167	0.9514
10	128	24	32	16	8	tanh	1.00E-07	4	0.1927	0.499	0.4995	0.1874	0.4995	0.8985	0.0657	0.9333	0.0667	0.9343
10	128	24	10	5	2	tanh	1.00E-07	4	0.1902	0.496	0.4981	0.1846	0.4981	0.8999	0.0629	0.9333	0.0667	0.9371
10	128	24	5	2	1	tanh	1.00E-07	4	0.1707	0.5018	0.5005	0.1602	0.5005	0.9106	0.0343	0.9667	0.0333	0.9657
10	128	24	5	3	1	tanh	1.00E-07	4	0.2	0.4946	0.4969	0.1963	0.4969	0.8944	0.0771	0.9167	0.0833	0.9229
10	128	24	50	20	10	tanh	1.00E-07	4	0.1878	0.5084	0.5036	0.1811	0.5036	0.9012	0.0571	0.95	0.05	0.9429
10	12	24	50	20	10	tanh	1.00E-07	4	0.1707	0.575	0.5143	0.1582	0.5143	0.9106	0.0286	1	0	0.9714
10	12	24	5	2	1	tanh	1.00E-07	4	0.1927	0.499	0.4995	0.1874	0.4995	0.8985	0.0657	0.9333	0.0667	0.9343
10	256	24	5	2	1	tanh	1.00E-07	4	0.1927	0.499	0.4995	0.1874	0.4995	0.8985	0.0657	0.9333	0.0667	0.9343
10	128	24	5	2	1	sigmoid	1.00E-07	4	0.1878	0.5269	0.5105	0.1804	0.5105	0.9012	0.0543	0.9667	0.0333	0.9457

10	128	24	5	2	1	hard_sigmoid	1.00E-07	4	0.1902	0.5293	0.5119	0.18332	0.5119	0.8999	0.0571	0.9667	0.0333	0.9429
10	128	24	5	2	1	exponential	1.00E-07	4	0.1902	0.5293	0.5119	0.18332	0.5119	0.8999	0.0571	0.9667	0.0333	0.9429
10	128	24	5	2	1	linear	1.00E-07	4	0.2049	0.4889	0.4929	0.20221	0.4929	0.8917	0.0857	0.9	0.1	0.9143
10	128	24	5	2	1	tanh	1.00E-07	3	0.2146	0.498	0.4986	0.21288	0.4986	0.8862	0.0971	0.9	0.1	0.9029
10	128	24	5	2	1	tanh	1.00E-07	2	0.3024	0.5012	0.5017	0.29887	0.5017	0.8352	0.22	0.7833	0.2167	0.7
10	128	24	5	2	1	tanh	1.00E-07	1	0.6122	0.4863	0.476	0.46133	0.476	0.6227	0.6686	0.2833	0.7167	0.3314
10	128	24	5	2	1	tanh	1.00E-07	5	0.1707	0.5325	0.5074	0.15922	0.5074	0.9106	0.0314	0.9833	0.0167	0.9686
10	128	24	5	2	1	linear	1.00E-06	4	0.2	0.521	0.5107	0.19511	0.5107	0.8944	0.0714	0.95	0.05	0.9286
10	128	24	5	2	1	tanh	1.00E-08	4	0.2	0.507	0.5038	0.19577	0.5038	0.8944	0.0743	0.9333	0.0667	0.9257
10	128	24	5	2	1	tanh	1.00E-09	4	0.1927	0.4745	0.4857	0.18866	0.4857	0.8985	0.0714	0.9	0.1	0.9286

10	128	24	5	2	1	tanh	1.00E-06	4	0.2	0.507	0.5038	0.1957	0.5038	0.8944	0.0743	0.9333	0.0667	0.9257

Appendix V: All results in K-means Model for Experiment 4.

two clusters method														
Tuning Parameters					**Evaluations**									
initialization	**n_init**	**max_iter**	**algorithm**	**RandomState**	**Accuracy**	**Precision**	**Recall**	**F1-score**	**ROC auc score**	**RMSE**	**TPR**	**TNR**	**FPR**	**FNR**
K-means++	5	1	auto	0	0.5114	0.5099	0.5867	0.3711	0.5867	0.699	0.6667	0.5067	0.4933	0.3333
K-means++	5	10	auto	0	0.5016	0.5093	0.5817	0.366	0.5817	0.706	0.6667	0.4966	0.5034	0.3333
K-means++	5	1	auto	42	0.4137	0.5107	0.5902	0.3218	0.5902	0.7657	0.7778	0.4027	0.5973	0.2222
K-means++	5	10	auto	42	0.43	0.5052	0.5447	0.3272	0.5447	0.755	0.6667	0.4228	0.5772	0.3333
K-means++	5	1	auto	1	0.3322	0.4855	0.3867	0.2632	0.3867	0.8172	0.4444	0.3289	0.6711	0.5556
K-means++	5	10	auto	1	0.4984	0.4907	0.4183	0.3491	0.4183	0.7083	0.3333	0.5034	0.4966	0.6667
K-means++	5	1	auto	2	0.5505	0.5122	0.6068	0.3913	0.6068	0.6705	0.6667	0.547	0.453	0.3333
K-means++	5	10	auto	2	0.5016	0.5093	0.5817	0.366	0.5817	0.706	0.6667	0.4966	0.5034	0.3333
K-means++	5	1	auto	3	0.4495	0.4878	0.3932	0.3247	0.3932	0.7419	0.3333	0.453	0.547	0.6667

K-means++	5	10	auto	3	0.4984	0.4907	0.4183	0.3491	0.4183	0.7083	0.3333	0.5034	0.4966	0.6667
K-means++	5	1	auto	4	0.4886	0.4901	0.4133	0.3443	0.4133	0.7151	0.3333	0.4933	0.5067	0.6667
K-means++	5	10	auto	4	0.4984	0.4907	0.4183	0.3491	0.4183	0.7083	0.3333	0.5034	0.4966	0.6667
K-means++	5	1	auto	5	0.6352	0.5118	0.5966	0.4272	0.5966	0.604	0.5556	0.6376	0.3624	0.4444
K-means++	5	10	auto	5	0.5016	0.5093	0.5817	0.366	0.5817	0.706	0.6667	0.4966	0.5034	0.3333
K-means++	5	1	auto	13	0.6189	0.4976	0.4804	0.4052	0.4804	0.6173	0.3333	0.6275	0.3725	0.6667
K-means++	5	10	auto	13	0.5016	0.5093	0.5817	0.366	0.5817	0.706	0.6667	0.4966	0.5034	0.3333
K-means++	5	1	auto	14	0.4267	0.4925	0.4353	0.3171	0.4353	0.7572	0.4444	0.4262	0.5738	0.5556
K-means++	5	10	auto	14	0.4984	0.4907	0.4183	0.3491	0.4183	0.7083	0.3333	0.5034	0.4966	0.6667
K-means++	5	1	auto	90	0.4365	0.4931	0.4403	0.3223	0.4403	0.7507	0.4444	0.4362	0.5638	0.5556
K-means++	5	10	auto	90	0.4984	0.4907	0.4183	0.3491	0.4183	0.7083	0.3333	0.5034	0.4966	0.6667
K-means++	5	1	auto	91	0.43	0.5052	0.5447	0.3272	0.5447	0.755	0.6667	0.4228	0.5772	0.3333
K-means++	5	10	auto	91	0.43	0.5052	0.5447	0.3272	0.5447	0.755	0.6667	0.4228	0.5772	0.3333
K-means++	5	1	auto	92	0.5147	0.4978	0.4806	0.3625	0.4806	0.6967	0.4444	0.5168	0.4832	0.5556
K-means++	5	10	auto	92	0.57	0.4948	0.4553	0.3831	0.4553	0.6557	0.3333	0.5772	0.4228	0.6667

K-means++	5	1	auto	200	0.645	0.4924	0.44	0.4089	0.44	0.5959	0.2222	0.6577	0.3423	0.7778
K-means++	5	10	auto	200	0.5016	0.5093	0.5817	0.366	0.5817	0.706	0.6667	0.4966	0.5034	0.3333
K-means++	5	1	auto	250	0.5049	0.4972	0.4756	0.3576	0.4756	0.7036	0.4444	0.5067	0.4933	0.5556
K-means++	5	10	auto	250	0.5016	0.5093	0.5817	0.366	0.5817	0.706	0.6667	0.4966	0.5034	0.3333
random	5	1	Full	5	0.5505	0.5122	0.6068	0.3913	0.6068	0.6705	0.6667	0.547	0.453	0.3333
random	5	10	elkan	5	0.5016	0.5093	0.5817	0.366	0.5817	0.706	0.6667	0.4966	0.5034	0.3333
random	10	10	auto	5	0.4495	0.4878	0.3932	0.3247	0.3932	0.7419	0.3333	0.453	0.547	0.6667

Appendix W: All results in HMM Model for Experiment 4.

two states method														
Tuning Parameters						**Evaluations**								
covariance_type	**min_covar**	**n_iter**	**algorithm**	**tol**	**Random State**	**Accuracy**	**Precision**	**Recall**	**F1-score**	**ROC auc score**	**RMSE**	**TPR**	**TNR**	**FPR**	**FNR**
spherical	0.0001	5000	viterbi	0.1	defaults	0.0065	0.0033	0.111	0.006	0.1111111	0.9967	0.222	0	1	0.7778
diag	0.0001	5000	viterbi	0.1	defaults	0.9935	0.9967	0.889	0.936	0.8888889	0.0807	0.778	1	0	0.2222
tied	0.0001	5000	viterbi	0.1	defaults	0.57	0.4948	0.455	0.383	0.4552573	0.6557	0.333	0.5772	0.42	0.6667
full	0.0001	5000	viterbi	0.1	defaults	0.9935	0.9967	0.889	0.936	0.8888889	0.0807	0.778	1	0	0.2222

spherical	0.0001	defaults	viterbi	defaults	defaults	0.0098	0.0658	0.113	0.01	0.112789	0.9951	0.222	0.0034	1	0.7778
diag	0.0001	defaults	viterbi	defaults	defaults	0.9902	0.9342	0.887	0.909	0.887211	0.0989	0.778	0.9966	0	0.2222
tied	0.0001	defaults	viterbi	defaults	defaults	0.57	0.4948	0.455	0.383	0.4552573	0.6557	0.333	0.5772	0.42	0.6667
full	0.0001	defaults	viterbi	defaults	defaults	0.9935	0.9967	0.889	0.936	0.8888889	0.0807	0.778	1	0	0.2222
spherical	0.0001	5000	map	0.1	defaults	0.0065	0.0033	0.111	0.006	0.1111111	0.9967	0.222	0	1	0.7778
diag	0.0001	5000	map	0.1	defaults	0.9935	0.9967	0.889	0.936	0.8888889	0.0807	0.778	1	0	0.2222
tied	0.0001	5000	map	0.1	defaults	0.557	0.494	0.449	0.377	0.4485459	0.6656	0.333	0.5638	0.44	0.6667
full	0.0001	5000	map	0.1	defaults	0.9935	0.9967	0.889	0.936	0.8888889	0.0807	0.778	1	0	0.2222
spherical	0.0001	defaults	map	defaults	defaults	0.9902	0.9342	0.887	0.909	0.887211	0.0989	0.778	0.9966	0	0.2222
diag	0.0001	defaults	map	defaults	defaults	0.0098	0.0658	0.113	0.01	0.112789	0.9951	0.222	0.0034	1	0.7778
tied	0.0001	defaults	map	defaults	defaults	0.443	0.506	0.551	0.334	0.5514541	0.7463	0.667	0.4362	0.56	0.3333
full	0.0001	defaults	map	defaults	defaults	0.0065	0.0033	0.111	0.006	0.1111111	0.9967	0.222	0	1	0.7778
spherical	0.0001	5000	viterbi	defaults	defaults	0.0065	0.0033	0.111	0.006	0.1111111	0.9967	0.222	0	1	0.7778
spherical	0.0001	5	viterbi	0.1	defaults	0.9349	0.6365	0.859	0.689	0.8586875	0.2552	0.778	0.9396	0.06	0.2222
spherical	0.0001	5	viterbi	0.1	42	0.0651	0.3635	0.141	0.063	0.1413125	0.9669	0.222	0.0604	0.94	0.7778

spherical	0.0001	5000	map	0.1	1400	0.0065	0.0033	0.111	0.006	0.1111111	0.9967	0.222	0	1	0.7778

Appendix X: All results in Auto-Encoder Model for Experiment 4.

Threshold Method																	
Tuning Parameters									**Evaluations**								
nb_epoch	**batch_size**	**input_dim**	**encoding_dim**	**hidden_dim1**	**hidden_dim2**	**activation**	**learning_rate**	**Threshold**	**Accuracy**	**Precision**	**Recall**	**F1-score**	**ROC aucscore**	**RMSE**	**TPR**	**TNR**	**FPR**	**FNR**
10	128	2	18	10	6	tanh	1.00E-07	4	0.9902	0.9342	0.8872	0.9093	0.8872	0.0989	0.7778	0.9966	0.0034	0.2222
50	128	2	18	10	6	tanh	1.00E-07	4	0.9837	0.9917	0.7222	0.8035	0.7222	0.1276	0.4444	1	0	0.5556
10	128	2	32	16	8	tanh	1.00E-07	4	0.9935	0.9967	0.8889	0.9358	0.8889	0.0807	0.7778	1	0	0.2222
10	128	2	10	5	2	tanh	1.00E-07	4	0.9935	0.9967	0.8889	0.9358	0.8889	0.0807	0.7778	1	0	0.2222
10	128	2	5	2	1	tanh	1.00E-07	4	0.987	0.8855	0.8855	0.8855	0.8855	0.1141	0.7778	0.9933	0.0067	0.2222
10	128	2	5	3	1	tanh	1.00E-07	4	0.9902	0.9342	0.8872	0.9093	0.8872	0.0989	0.7778	0.9966	0.0034	0.2222
10	128	2	50	20	10	tanh	1.00E-07	4	0.987	0.9934	0.7778	0.8538	0.7778	0.1141	0.5556	1	0	0.4444
10	12	2	50	20	10	tanh	1.00E-07	4	0.9837	0.9917	0.7222	0.8035	0.7222	0.1276	0.4444	1	0	0.5556
10	12	2	5	2	1	tanh	1.00E-07	4	0.99023	0.93416	0.88721	0.90925	0.88721	0.09885	0.77778	0.99664	0.00336	0.22222

10	256	2	5	2	1	tanh	1.00E-07	4	0.99 02	0.93 42	0.8 872	0.9 093	0.8 872	0.0 989	0.7 778	0.9 966	0.0 034	0.2 222
10	128	2	5	2	1	sigmo id	1.00E-07	4	0.99 02	0.93 42	0.8 872	0.9 093	0.8 872	0.0 989	0.7 778	0.9 966	0.0 034	0.2 222
10	128	2	5	2	1	hard_ sigmo id	1.00E-07	4	0.99 02	0.93 42	0.8 872	0.9 093	0.8 872	0.0 989	0.7 778	0.9 966	0.0 034	0.2 222
10	128	2	5	2	1	expon ential	1.00E-07	4	0.99 02	0.93 42	0.8 872	0.9 093	0.8 872	0.0 989	0.7 778	0.9 966	0.0 034	0.2 222
10	128	2	5	2	1	linear	1.00E-07	4	0.97 72	0.86 51	0.6 65	0.7 249	0.6 65	0.1 51	0.3 333	0.9 966	0.0 034	0.6 667
10	128	2	5	2	1	tanh	1.00E-07	3	0.98 7	0.88 55	0.8 855	0.8 855	0.8 855	0.1 141	0.7 778	0.9 933	0.0 067	0.2 222
10	128	2	5	2	1	tanh	1.00E-07	2	0.97 07	0.74 66	0.8 771	0.7 967	0.8 771	0.1 712	0.7 778	0.9 765	0.0 235	0.2 222
10	128	2	5	2	1	tanh	1.00E-07	1	0.86 32	0.57 06	0.8 218	0.5 874	0.8 218	0.3 699	0.7 778	0.8 658	0.1 342	0.2 222
10	128	2	5	2	1	tanh	1.00E-07	5	0.99 02	0.93 42	0.8 872	0.9 093	0.8 872	0.0 989	0.7 778	0.9 966	0.0 034	0.2 222
10	128	2	5	2	1	linear	1.00E-06	4	0.97 72	0.86 51	0.6 65	0.7 249	0.6 65	0.1 51	0.3 333	0.9 966	0.0 034	0.6 667
10	128	2	5	2	1	tanh	1.00E-08	4	0.99 02	0.93 42	0.8 872	0.9 093	0.8 872	0.0 989	0.7 778	0.9 966	0.0 034	0.2 222
10	128	2	5	2	1	tanh	1.00E-09	4	0.99 02	0.93 42	0.8 872	0.9 093	0.8 872	0.0 989	0.7 778	0.9 966	0.0 034	0.2 222
10	128	2	5	2	1	tanh	1.00E-06	4	0.99 02	0.93 42	0.8 872	0.9 093	0.8 872	0.0 989	0.7 778	0.9 966	0.0 034	0.2 222

Appendix Y: All results in K-means Model for Experiment 5.

creditcard dataset

two clusters method														
Tuning Parameters					**Evaluations**									
initialization	**n_init**	**max_iter**	**algorithm**	**RandomState**	**Accuracy**	**Precision**	**Recall**	**F1-score**	**ROC auc score**	**RMSE**	**TPR**	**TNR**	**FPR**	**FNR**
K-means++	5	1	auto	0	0.42	0.4517	0.3667	0.3336	0.3667	0.7616	0.3	0.4333	0.5667	0.7
K-means++	5	10	auto	0	0.57	0.5303	0.5833	0.4608	0.5833	0.6557	0.6	0.5667	0.4333	0.4
K-means++	5	1	auto	42	0.43	0.5189	0.55	0.3777	0.55	0.755	0.7	0.4	0.6	0.3
K-means++	5	10	auto	42	0.54	0.524	0.5667	0.4415	0.5667	0.6782	0.6	0.5333	0.4667	0.4
K-means++	5	1	auto	1	0.45	0.5061	0.5167	0.3828	0.5167	0.7416	0.6	0.4333	0.5667	0.4
K-means++	5	10	auto	1	0.6	0.5684	0.6889	0.504	0.6889	0.6325	0.8	0.5778	0.4222	0.2
K-means++	5	1	auto	2	0.38	0.49132	0.47778	0.33505	0.47778	0.7874	0.6	0.35556	0.64444	0.4
K-means++	5	10	auto	2	0.41	0.4495	0.3611	0.3276	0.3611	0.7681	0.3	0.4222	0.5778	0.7
K-means++	5	1	auto	3	0.36	0.4696	0.4222	0.3136	0.4222	0.8	0.5	0.3444	0.6556	0.5
K-means++	5	10	auto	3	0.55	0.542	0.6167	0.4591	0.6167	0.6708	0.7	0.5333	0.4667	0.3
K-means++	5	1	auto	4	0.61	0.5064	0.5167	0.4577	0.5167	0.6245	0.4	0.6333	0.3667	0.6
K-means++	5	10	auto	4	0.58	0.5325	0.5889	0.4673	0.5889	0.6481	0.6	0.5778	0.4222	0.4

K-means++	5	1	auto	5	0.56	0.4959	0.4889	0.4283	0.4889	0.6633	0.4	0.5778	0.4222	0.6
K-means++	5	10	auto	5	0.46	0.476	0.4333	0.3689	0.4333	0.7348	0.4	0.4667	0.5333	0.6
K-means++	5	1	auto	13	0.54	0.4919	0.4778	0.4165	0.4778	0.6782	0.4	0.5556	0.4444	0.6
K-means++	5	10	auto	13	0.45	0.4739	0.4278	0.3628	0.4278	0.7416	0.4	0.4556	0.5444	0.6
K-means++	5	1	auto	14	0.43	0.4697	0.4167	0.3504	0.4167	0.755	0.4	0.4333	0.5667	0.6
K-means++	5	10	auto	14	0.54	0.524	0.5667	0.4415	0.5667	0.6782	0.6	0.5333	0.4667	0.4
K-means++	5	1	auto	90	0.53	0.5542	0.65	0.4555	0.65	0.6856	0.8	0.5	0.5	0.2
K-means++	5	10	auto	90	0.57	0.5462	0.6278	0.4725	0.6278	0.6557	0.7	0.5556	0.4444	0.3
K-means++	5	1	auto	91	0.47	0.5434	0.6167	0.4137	0.6167	0.728	0.8	0.4333	0.5667	0.2
K-means++	5	10	auto	91	0.6	0.5528	0.6444	0.4926	0.6444	0.6325	0.7	0.5889	0.4111	0.3
K-means++	5	1	auto	92	0.63	0.511	0.5278	0.4695	0.5278	0.6083	0.4	0.6556	0.3444	0.6
K-means++	5	10	auto	92	0.59	0.5505	0.6389	0.4859	0.6389	0.6403	0.7	0.5778	0.4222	0.3
K-means++	5	1	auto	200	0.44	0.5208	0.5556	0.3846	0.5556	0.7483	0.7	0.4111	0.5889	0.3
K-means++	5	10	auto	200	0.56	0.5441	0.6222	0.4658	0.6222	0.6633	0.7	0.5444	0.4556	0.3
K-means++	5	1	auto	250	0.45	0.5227	0.5611	0.3915	0.5611	0.7416	0.7	0.4222	0.5778	0.3

K-means++	5	10	auto	250	0.44	0.4878	0.4667	0.3668	0.4667	0.7483	0.5	0.4333	0.5667	0.5
random	5	1	Full	5	0.56	0.4959	0.4889	0.4283	0.4889	0.6633	0.4	0.5778	0.4222	0.6
random	5	10	elkan	5	0.46	0.476	0.4333	0.3689	0.4333	0.7348	0.4	0.4667	0.5333	0.6
random	10	10	auto	5	0.54	0.4919	0.4778	0.4165	0.4778	0.6782	0.4	0.5556	0.4444	0.6

Appendix Z: All results in HMM Model for Experiment 5.

two states method														
Tuning Parameters						**Evaluations**								
covariance_type	**min_covar**	**n_iter**	**algorithm**	**tol**	**Random State**	**Accuracy**	**Precision**	**Recall**	**F1-score**	**ROC auc score**	**RMSE**	**TPR**	**TNR**	**FPR**	**FNR**
spherical	0.0001	5000	viterbi	0.1	defaults	0.59	0.5347	0.594	0.474	0.5944444	0.6403	0.6	0.5889	0.41	0.4
diag	0.0001	5000	viterbi	0.1	defaults	0.57	0.5303	0.5833	0.4608	0.58333333	0.65574	0.6	0.56667	0.433	0.4
tied	0.0001	5000	viterbi	0.1	defaults	0.56	0.5282	0.578	0.454	0.5777778	0.6633	0.6	0.5556	0.44	0.4
full	0.0001	defaults	viterbi	0.1	defaults	0.56	0.5282	0.578	0.454	0.5777778	0.6633	0.6	0.5556	0.44	0.4
spherical	0.0001	defaults	viterbi	defaults	defaults	0.61	0.5393	0.606	0.487	0.6055556	0.6245	0.6	0.6111	0.39	0.4
diag	0.0001	defaults	viterbi	defaults	defaults	0.57	0.5303	0.583	0.461	0.5833333	0.6557	0.6	0.5667	0.43	0.4
tied	0.0001	defaults	viterbi	defaults	defaults	0.53	0.506	0.517	0.423	0.5166667	0.6856	0.5	0.5333	0.47	0.5

full	0.0001	defaults	viterbi	defaults	defaults	0.53	0.506	0.517	0.423	0.516667	0.6856	0.5	0.5333	0.47	0.5
spherical	0.0001	5000	map	0.1	defaults	0.6	0.5369	0.6	0.48	0.6	0.6325	0.6	0.6	0.4	0.4
diag	0.0001	5000	map	0.1	defaults	0.56	0.5282	0.578	0.454	0.577778	0.6633	0.6	0.5556	0.44	0.4
tied	0.0001	5000	map	0.1	defaults	0.57	0.5303	0.583	0.461	0.583333	0.6557	0.6	0.5667	0.43	0.4
full	0.0001	defaults	map	0.1	defaults	0.56	0.5282	0.578	0.454	0.577778	0.6633	0.6	0.5556	0.44	0.4
spherical	0.0001	defaults	map	defaults	defaults	0.61	0.5393	0.606	0.487	0.6055556	0.6245	0.6	0.6111	0.39	0.4
diag	0.0001	defaults	map	defaults	defaults	0.56	0.5282	0.578	0.454	0.577778	0.6633	0.6	0.5556	0.44	0.4
tied	0.0001	defaults	map	defaults	defaults	0.54	0.5081	0.522	0.43	0.522222	0.6782	0.5	0.5444	0.46	0.5
full	0.0001	defaults	map	defaults	defaults	0.54	0.5081	0.522	0.43	0.522222	0.6782	0.5	0.5444	0.46	0.5
spherical	0.0001	5000	viterbi	defaults	defaults	0.61	0.5393	0.606	0.487	0.6055556	0.6245	0.6	0.6111	0.39	0.4
spherical	0.0001	5	viterbi	0.1	defaults	0.59	0.5347	0.594	0.474	0.5944444	0.6403	0.6	0.5889	0.41	0.4
spherical	0.0001	5	viterbi	0.1	42	0.4	0.4631	0.4	0.332	0.4	0.7746	0.4	0.4	0.6	0.6

Appendix AA: All results in Auto-Encoder Model for Experiment 5.

Threshold Method	
Tuning Parameters	**Evaluations**

nb_epoch	batch_size	input_dim	encoding_dim	hidden_dim1	hidden_dim2	activation	learning_rate	Threshold	Accuracy	Precision	Recall	F1-score	ROC auc score	RMSE	TPR	TNR	FPR	FNR
10	128	11	18	10	6	tanh	1.00E-07	4	0.92	0.9592	0.6	0.6454	0.6	0.2828	0.2	1	0	0.8
50	128	11	18	10	6	tanh	1.00E-07	4	0.91	0.9545	0.55	0.5671	0.55	0.3	0.1	1	0	0.9
10	128	11	32	16	8	tanh	1.00E-07	4	0.92	0.9592	0.6	0.6454	0.6	0.2828	0.2	1	0	0.8
10	128	11	10	5	2	tanh	1.00E-07	4	0.91	0.7535	0.6833	0.7107	0.6833	0.3	0.4	0.9667	0.0333	0.6
10	128	11	5	2	1	tanh	1.00E-07	4	0.9	0.7174	0.6778	0.6947	0.6778	0.3162	0.4	0.9556	0.0444	0.6
10	128	11	5	3	1	tanh	1.00E-07	4	0.91	0.7535	0.6833	0.7107	0.6833	0.3	0.4	0.9667	0.0333	0.6
10	128	11	50	20	10	tanh	1.00E-07	4	0.92	0.9592	0.6	0.6454	0.6	0.2828	0.2	1	0	0.8

10	12	11	50	20	10	tanh	1.00E-07	4	0.92	0.9592	0.6	0.6454	0.6	0.2828	0.2	1	0	0.8
10	12	11	5	2	1	tanh	1.00E-07	4	0.92	0.8385	0.6444	0.6928	0.6444	0.2828	0.3	0.9889	0.0111	0.7
10	256	11	5	2	1	tanh	1.00E-07	4	0.9	0.7174	0.6778	0.6947	0.6778	0.3162	0.4	0.9556	0.0444	0.6
10	128	11	5	2	1	sigmoid	1.00E-07	4	0.88	0.6667	0.6667	0.6667	0.6667	0.3464	0.4	0.9333	0.0667	0.6
10	128	11	5	2	1	hard_sigmoid	1.00E-07	4	0.88	0.6667	0.6667	0.6667	0.6667	0.3464	0.4	0.9333	0.0667	0.6
10	128	11	5	2	1	exponential	1.00E-07	4	0.1	0.05	0.5	0.0909	0.5	0.9487	1	0	1	0
10	128	11	5	2	1	linear	1.00E-07	4	0.91	0.7503	0.7278	0.7383	0.7278	0.3	0.5	0.9556	0.0444	0.5
10	128	11	5	2	1	tanh	1.00E-07	3	0.85	0.6373	0.6944	0.6571	0.6944	0.3873	0.5	0.8889	0.1111	0.5
10	128	11	5	2	1	tanh	1.00E-07	2	0.82	0.6084	0.6778	0.6262	0.6778	0.4243	0.5	0.8556	0.1444	0.5
10	128	11	5	2	1	tanh	1.00E-07	1	0.58	0.5641	0.6778	0.49	0.6778	0.6481	0.8	0.5556	0.4444	0.2

10	128	11	5	2	1	tanh	1.00E-07	5	0.91	0.7535	0.6833	0.71077	0.6833	0.3	0.4	0.96667	0.0333	0
10	128	11	5	2	1	tanh	1.00E-06	4	0.9	0.7174	0.6778	0.69477	0.6778	0.3162	0.4	0.95556	0.0444	0.6
10	128	11	5	2	1	tanh	1.00E-08	4	0.9	0.7174	0.6778	0.69477	0.6778	0.3162	0.4	0.95556	0.0444	0.6
10	128	11	5	2	1	tanh	1.00E-09	4	0.92	0.8385	0.6444	0.69288	0.6444	0.2828	0.3	0.98889	0.0111	0.7
10	128	11	5	2	1	linear	1.00E-06	4	0.9	0.7128	0.6333	0.66033	0.6333	0.3162	0.3	0.96667	0.0333	0.7

Appendix BB: All results in K-means Model for Experiment 6.

two clusters method														
Tuning Parameters					**Evaluations**									
initialization	**n_init**	**max_iter**	**algorithm**	**RandomState**	**Accuracy**	**Precision**	**Recall**	**F1-score**	**ROC auc score**	**RMSE**	**TPR**	**TNR**	**FPR**	**FNR**
K-means++	5	1	auto	0	0.9883	0.4952	0.4989	0.4971	0.4989	0.1081	0	0.9979	0.0021	1
K-means++	5	10	auto	0	0.9883	0.4952	0.4989	0.4971	0.4989	0.1081	0	0.9979	0.0021	1
K-means++	5	1	auto	42	0.9883	0.4952	0.4989	0.497	0.4989	0.1084	0	0.9978	0.0022	1
K-means++	5	10	auto	42	0.9883	0.4952	0.4989	0.4971	0.4989	0.1081	0	0.9979	0.0021	1

K-means++	5	1	auto	1	0.9883	0.4952	0.4989	0.4971	0.4989	0.1081	0	0.9979	0.0021	1
K-means++	5	10	auto	1	0.9883	0.4952	0.4989	0.4971	0.4989	0.1081	0	0.9979	0.0021	1
K-means++	5	1	auto	2	0.98869	0.49521	0.49912	0.49716	0.49912	0.10635	0	0.99824	0.00176	1
K-means++	5	10	auto	2	0.9883	0.4952	0.4989	0.4971	0.4989	0.1081	0	0.9979	0.0021	1
K-means++	5	1	auto	3	0.9883	0.4952	0.4989	0.4971	0.4989	0.1081	0	0.9979	0.0021	1
K-means++	5	10	auto	3	0.9883	0.4952	0.4989	0.4971	0.4989	0.1081	0	0.9979	0.0021	1
K-means++	5	1	auto	4	0.9883	0.4952	0.4989	0.4971	0.4989	0.1081	0	0.9979	0.0021	1
K-means++	5	10	auto	4	0.9883	0.4952	0.4989	0.4971	0.4989	0.1081	0	0.9979	0.0021	1
K-means++	5	1	auto	5	0.9883	0.4952	0.4989	0.4971	0.4989	0.1081	0	0.9979	0.0021	1
K-means++	5	10	auto	5	0.9883	0.4952	0.4989	0.4971	0.4989	0.1081	0	0.9979	0.0021	1
K-means++	5	1	auto	13	0.9883	0.4952	0.4989	0.4971	0.4989	0.1081	0	0.9979	0.0021	1
K-means++	5	10	auto	13	0.9883	0.4952	0.4989	0.4971	0.4989	0.1081	0	0.9979	0.0021	1
K-means++	5	1	auto	14	0.9883	0.4952	0.4989	0.4971	0.4989	0.1081	0	0.9979	0.0021	1
K-means++	5	10	auto	14	0.4464	0.4895	0.2254	0.3086	0.2254	0.744	0	0.4507	0.5493	1
K-means++	5	1	auto	90	0.9883	0.4952	0.4989	0.4971	0.4989	0.1081	0	0.9979	0.0021	1

K-means++	5	10	auto	90	0.9883	0.4952	0.4989	0.4971	0.4989	0.1081	0	0.9979	0.0021	1
K-means++	5	1	auto	91	0.9883	0.4952	0.4989	0.4971	0.4989	0.1081	0	0.9979	0.0021	1
K-means++	5	10	auto	91	0.9883	0.4952	0.4989	0.4971	0.4989	0.1081	0	0.9979	0.0021	1
K-means++	5	1	auto	92	0.9883	0.4952	0.4989	0.4971	0.4989	0.1081	0	0.9979	0.0021	1
K-means++	5	10	auto	92	0.9883	0.4952	0.4989	0.4971	0.4989	0.1081	0	0.9979	0.0021	1
K-means++	5	1	auto	200	0.0122	0.5048	0.5013	0.0122	0.5013	0.9939	1	0.0027	0.9973	0
K-means++	5	10	auto	200	0.0117	0.5048	0.5011	0.0116	0.5011	0.9941	1	0.0021	0.9979	0
K-means++	5	1	auto	250	0.9896	0.4952	0.4996	0.4974	0.4996	0.1019	0	0.9992	0.0008	1
K-means++	5	10	auto	250	0.9883	0.4952	0.4989	0.4971	0.4989	0.1081	0	0.9979	0.0021	1
random	5	1	auto	5	0.4197	0.4889	0.2119	0.2956	0.2119	0.7618	0	0.4237	0.5763	1
random	5	10	Full	5	0.9883	0.4952	0.4989	0.497	0.4989	0.1084	0	0.9978	0.0022	1
random	5	10	elkan	5	0.9883	0.4952	0.4989	0.497	0.4989	0.1084	0	0.9978	0.0022	1
random	1	1	auto	5	0.4696	0.5089	0.7322	0.3346	0.7322	0.7283	1	0.4645	0.5355	0
random	2	10	auto	5	0.9895	0.4952	0.4995	0.4974	0.4995	0.1025	0	0.9991	0.0009	1

Appendix CC: All results in HMM Model for Experiment 6.

two states method															
Tuning Parameters						**Evaluations**									
covariance_type	**min_covar**	**n_iter**	**algorithm**	**tol**	**Random State**	**Accuracy**	**Precision**	**Recall**	**F1-score**	**ROC auc score**	**RMSE**	**TPR**	**TNR**	**FPR**	**FNR**
spherical	0.0001	5000	viterbi	0.1	defaults	0.4021	0.5079	0.698	0.299	0.6981617	0.7732	1	0.3963	0.6	0
diag	0.0001	5000	viterbi	0.1	defaults	0.9863	0.7053	0.993	0.788	0.9930673	0.1172	1	0.9861	0.01	0
tied	0.0001	5000	viterbi	0.1	defaults	0.5037	0.5035	0.592	0.346	0.5918972	0.7045	0.682	0.502	0.5	0.3182
full	0.0001	defaults	viterbi	0.1	defaults	0.9863	0.7059	0.993	0.788	0.9930987	0.1169	1	0.9862	0.01	0
spherical	0.0001	defaults	viterbi	defaults	defaults	0.4017	0.5079	0.698	0.299	0.6979422	0.7735	1	0.3959	0.6	0
diag	0.0001	defaults	viterbi	defaults	defaults	0.9863	0.7053	0.993	0.788	0.9930673	0.1172	1	0.9861	0.01	0
tied	0.0001	defaults	viterbi	defaults	defaults	0.4963	0.4965	0.408	0.337	0.4081028	0.7097	0.318	0.498	0.5	0.6818
full	0.0001	defaults	viterbi	defaults	defaults	0.9863	0.7059	0.993	0.788	0.9930987	0.1169	1	0.9862	0.01	0
spherical	0.0001	5000	map	0.1	defaults	0.4021	0.5079	0.698	0.299	0.6981617	0.7732	1	0.3963	0.6	0
diag	0.0001	5000	map	0.1	defaults	0.9863	0.7053	0.993	0.788	0.9930673	0.1172	1	0.9861	0.01	0
tied	0.0001	5000	map	0.1	defaults	0.5038	0.5035	0.592	0.346	0.5919286	0.7044	0.682	0.502	0.5	0.3182
full	0.0001	defaults	map	0.1	defaults	0.9863	0.7059	0.993	0.788	0.9930987	0.1169	1	0.9862	0.01	0

spherical	0.0001	defaults	map	defaults	defaults	0.4016	0.5079	0.698	0.299	0.6979108	0.7736	1	0.3958	0.6	0
diag	0.0001	defaults	map	defaults	defaults	0.9863	0.7053	0.993	0.788	0.9930673	0.11772	1	0.9861	0.01	0
tied	0.0001	defaults	map	defaults	defaults	0.4962	0.4965	0.408	0.337	0.4080714	0.7098	0.318	0.498	0.5	0.6818
full	0.0001	defaults	map	defaults	defaults	0.9863	0.7059	0.993	0.788	0.9930987	0.1169	1	0.9862	0.01	0
spherical	0.0001	5000	viterbi	defaults	defaults	0.4017	0.5079	0.698	0.299	0.6979422	0.7735	1	0.3959	0.6	0
spherical	0.0001	5	viterbi	0.1	defaults	0.4448	0.5083	0.716	0.322	0.7164973	0.7451	0.994	0.4395	0.56	0.0065
spherical	0.0001	5	viterbi	0.1	42	0.5554	0.4917	0.284	0.357	0.2835968	0.6668	0.006	0.5607	0.44	0.9935

Appendix DD: All results in Auto-Encoder Model for Experiment 6.

Threshold Method																	
Tuning Parameters									**Evaluations**								
nb_epoch	**batch_size**	**input_dim**	**encoding_dim**	**hidden_dim1**	**hidden_dim2**	**activation**	**learning_rate**	**Threshold**	**Accuracy**	**Precision**	**Recall**	**F1-score**	**ROC auc score**	**RMSE**	**TPR**	**TNR**	**FPR**	**FNR**
10	128	4	18	10	6	tanh	1.00E-07	4	0.9963	0.8598	0.9981	0.9175	0.9981	0.0611	1	0.9962	0.0038	0

50	128	4	18	10	6	tanh	1.00E-07	4	0.9963	0.8598	0.9981	0.9175	0.9981	0.0611	1	0.9962	0.0038	0
10	128	4	32	16	8	tanh	1.00E-07	4	0.9963	0.8598	0.9981	0.9175	0.9981	0.0611	1	0.9962	0.0038	0
10	128	4	10	5	2	tanh	1.00E-07	4	0.9963	0.8598	0.9981	0.9175	0.9981	0.0611	1	0.9962	0.0038	0
10	128	4	5	2	1	tanh	1.00E-07	4	0.9963	0.8598	0.9981	0.9175	0.9981	0.0611	1	0.9962	0.0038	0
10	128	4	5	3	1	tanh	1.00E-07	4	0.9963	0.8598	0.9981	0.9175	0.9981	0.0611	1	0.9962	0.0038	0
10	128	4	50	20	10	tanh	1.00E-07	4	0.9963	0.8598	0.9981	0.9175	0.9981	0.0611	1	0.9962	0.0038	0
10	12	4	50	20	10	tanh	1.00E-07	4	0.9963	0.8598	0.9981	0.9175	0.9981	0.0611	1	0.9962	0.0038	0
10	12	4	5	2	1	tanh	1.00E-07	4	0.9963	0.8598	0.9981	0.9175	0.9981	0.0611	1	0.9962	0.0038	0
10	256	4	5	2	1	tanh	1.00E-07	4	0.9961	0.8548	0.998	0.91411	0.998	0.0626	1	0.996	0.004	0
10	128	4	5	2	1	sigmoid	1.00E-07	4	0.9941	0.8092	0.997	0.88066	0.997	0.0768	1	0.994	0.006	0

10	128	4	5	2	1	hard_sigmoid	1.00E-07	4	0.9939	0.8056	0.9969	0.8778	0.9969	0.078	1	0.9939	0.0061	0
10	128	4	5	2	1	exponential	1.00E-07	4	0.9939	0.8056	0.9969	0.8778	0.9969	0.078	1	0.9939	0.0061	0
10	128	4	5	2	1	linear	1.00E-07	4	0.9904	0.4952	0.5	0.4976	0.5	0.0978	0	1	0	1
10	128	4	5	2	1	tanh	1.00E-07	3	0.996	0.8516	0.998	0.9118	0.998	0.0636	1	0.9959	0.0041	0
10	128	4	5	2	1	tanh	1.00E-07	2	0.9956	0.8422	0.9978	0.9052	0.9978	0.0664	1	0.9955	0.0045	0
10	128	4	5	2	1	tanh	1.00E-07	1	0.9935	0.7984	0.9967	0.8721	0.9967	0.0804	1	0.9935	0	0
10	128	4	5	2	1	tanh	1.00E-07	5	0.9966	0.8702	0.9983	0.9246	0.9983	0.0579	1	0.9966	0.0034	0
10	128	4	5	2	1	linear	1.00E-06	4	0.9904	0.4952	0.5	0.4976	0.5	0.0978	0	1	0	1
10	128	4	5	2	1	tanh	1.00E-08	4	0.9963	0.8598	0.9981	0.9175	0.9981	0.0611	1	0.9962	0.0038	0
10	128	4	5	2	1	tanh	1.00E-09	4	0.9963	0.8598	0.9981	0.9175	0.9981	0.0611	1	0.9962	0.0038	0

10	128	4	5	2	1	tanh	1.00E-06	4	0.9963	0.8598	0.9981	0.9175	0.9981	0.0611	1	0.9962	0.0038	0

Appendix EE: All results in K-means Model for Experiment 7.

two clusters method														
Tuning Parameters					**Evaluations**									
initialization	**n_init**	**max_iter**	**algorithm**	**RandomState**	**Accuracy**	**Precision**	**Recall**	**F1-score**	**ROC auc score**	**RMSE**	**TPR**	**TNR**	**FPR**	**FNR**
K-means++	5	1	auto	0	0.3825	0.4779	0.4339	0.3228	0.4339	0.7858	0.4962	0.3717	0.6283	0.5038
K-means++	5	10	auto	0	0.3358	0.4755	0.4317	0.2948	0.4317	0.815	0.5475	0.3158	0.6842	0.4525
K-means++	5	1	auto	42	0.4204	0.4777	0.4308	0.3427	0.4308	0.7613	0.4434	0.4182	0.5818	0.5566
K-means++	5	10	auto	42	0.3111	0.4753	0.4348	0.2793	0.4348	0.83	0.5844	0.2852	0.7148	0.4156
K-means++	5	1	auto	1	0.4847	0.481	0.4399	0.3756	0.4399	0.7178	0.3856	0.4941	0.5059	0.6144
K-means++	5	10	auto	1	0.6504	0.524	0.5686	0.4828	0.5686	0.5913	0.4696	0.6675	0.3325	0.5304
K-means++	5	1	auto	2	0.539	0.4862	0.4571	0.4031	0.4571	0.679	0.358	0.5561	0.4439	0.642
K-means++	5	10	auto	2	0.3261	0.4759	0.4341	0.289	0.4341	0.8209	0.5646	0.3036	0.6964	0.4354
K-means++	5	1	auto	3	0.3962	0.478	0.4333	0.3303	0.4333	0.7771	0.4781	0.3884	0.6116	0.5219
K-means++	5	10	auto	3	0.3322	0.4758	0.433	0.2927	0.433	0.8172	0.5548	0.3112	0.6888	0.4452

K-means++	5	1	auto	4	0.7663	0.5262	0.5527	0.5213	0.5527	0.4835	0.2946	0.8109	0.1891	0.7054
K-means++	5	10	auto	4	0.6977	0.5248	0.5642	0.5008	0.5642	0.5498	0.4028	0.7256	0.2744	0.5972
K-means++	5	1	auto	5	0.3414	0.4805	0.4455	0.3002	0.4455	0.8115	0.5714	0.3197	0.6803	0.4286
K-means++	5	10	auto	5	0.331	0.4758	0.4331	0.2919	0.4331	0.8179	0.5565	0.3096	0.6904	0.4435
K-means++	5	1	auto	13	0.312	0.4784	0.4431	0.2809	0.4431	0.8294	0.6015	0.2847	0.7153	0.3985
K-means++	5	10	auto	13	0.3318	0.4757	0.4328	0.2924	0.4328	0.8175	0.555	0.3106	0.6894	0.445
K-means++	5	1	auto	14	0.2743	0.478	0.4483	0.2551	0.4483	0.8519	0.6586	0.238	0.762	0.3414
K-means++	5	10	auto	14	0.3096	0.4754	0.4351	0.2783	0.4351	0.8309	0.5868	0.2834	0.7166	0.4132
K-means++	5	1	auto	90	0.3783	0.4804	0.4417	0.3218	0.4417	0.7885	0.5182	0.3651	0.6349	0.4818
K-means++	5	10	auto	90	0.3228	0.4754	0.4332	0.2867	0.4332	0.8229	0.5667	0.2997	0.7003	0.4333
K-means++	5	1	auto	91	0.4382	0.4786	0.433	0.3521	0.433	0.7496	0.4267	0.4393	0.5607	0.5733
K-means++	5	10	auto	91	0.3379	0.4758	0.4323	0.2962	0.4323	0.8137	0.5463	0.3182	0.6818	0.4537
K-means++	5	1	auto	92	0.56252	0.51856	0.55811	0.4405	0.55811	0.66142	0.55278	0.56344	0.43656	0.44722
K-means++	5	10	auto	92	0.6629	0.5243	0.5679	0.4879	0.5679	0.5806	0.4531	0.6828	0.3172	0.5469
K-means++	5	1	auto	200	0.3376	0.476	0.4329	0.2961	0.4329	0.8139	0.5481	0.3177	0.6823	0.4519

K-means++	5	10	auto	200	0.3345	0.4757	0.4325	0.2941	0.4325	0.8158	0.5508	0.3141	0.6859	0.4492
K-means++	5	1	auto	250	0.9055	0.5207	0.5028	0.4894	0.5028	0.3073	0.0159	0.9897	0.0103	0.9841
K-means++	5	10	auto	250	0.3152	0.4754	0.4342	0.2819	0.4342	0.8275	0.5781	0.2904	0.7096	0.4219
random	5	1	auto	90	0.2121	0.474	0.4535	0.2066	0.4535	0.8876	0.7453	0.1617	0.8383	0.2547
random	5	10	Full	5	0.2121	0.474	0.4535	0.2066	0.4535	0.8876	0.7453	0.1617	0.8383	0.2547
random	5	10	elkan	5	0.9055	0.5207	0.5028	0.4894	0.5028	0.3073	0.0159	0.9897	0.0103	0.9841
random	20	1	auto	5	0.2121	0.474	0.4535	0.2066	0.4535	0.8876	0.7453	0.1617	0.8383	0.2547

Appendix FF: All results in HMM Model for Experiment 7.

two states method														
Tuning Parameters						**Evaluations**								
covariance_type	**min_covar**	**n_iter**	**algorithm**	**tol**	**Random State**	**Accuracy**	**Precision**	**Recall**	**F1-score**	**ROC auc score**	**RMSE**	**TPR**	**TNR**	**FPR**	**FNR**
spherical	0.0001	5000	viterbi	0.1	defaults	0.1075	0.4766	0.492	0.1105	0.4917639	0.9447	0.956	0.0272	0.97	0.0437
diag	0.0001	5000	viterbi	0.1	defaults	0.1472	0.4786	0.48	0.1147	0.4797749	0.9235	0.882	0.0777	0.92	0.1182
tied	0.0001	5000	viterbi	0.1	defaults										
full	0.0001	defaults	viterbi	0.1	defaults										
spherical	0.0001	defaults	viterbi	defaults	defaults	0.8871	0.5287	0.513	0.512	0.5129402	0.3359	0.061	0.9653	0.03	0.9394

diag	0.0001	defaults	viterbi	defaults	defaults	0.8528	0.5214	0.52	0.521	0.5202251	0.3837	0.118	0.9223	0.08	0.8818
tied	0.0001	defaults	viterbi	defaults	defaults	0.7694	0.5197	0.538	0.514	0.5381352	0.4802	0.259	0.8178	0.18	0.7415
full	0.0001	defaults	viterbi	defaults	defaults										
spherical	0.0001	5000	map	0.1	defaults	0.1075	0.4766	0.492	0.105	0.4917639	0.9447	0.956	0.0272	0.97	0.0437
diag	0.0001	5000	map	0.1	defaults	0.8528	0.5214	0.52	0.521	0.5202251	0.3837	0.118	0.9223	0.08	0.8818
tied	0.0001	5000	map	0.1	defaults	0.7434	0.5211	0.546	0.51	0.5464606	0.5066	0.308	0.7845	0.22	0.6916
full	0.0001	defaults	map	0.1	defaults										
spherical	0.0001	defaults	map	defaults	defaults	0.1125	0.4712	0.487	0.11	0.4872218	0.9421	0.94	0.0342	0.97	0.0597
diag	0.0001	defaults	map	defaults	defaults	0.8528	0.5214	0.52	0.521	0.5202251	0.3837	0.118	0.9223	0.08	0.8818
tied	0.0001	defaults	map	defaults	defaults	0.7787	0.5171	0.531	0.513	0.531189	0.4704	0.232	0.8304	0.17	0.768
full	0.0001	defaults	map	defaults	defaults										
spherical	0.0001	5000	viterbi	defaults	defaults	0.1075	0.4766	0.492	0.105	0.4917639	0.9447	0.956	0.0272	0.97	0
spherical	0.0001	5	viterbi	0.1	defaults	0.8342	0.5288	0.536	0.531	0.5356138	0.4072	0.175	0.8965	0.1	0.8253
spherical	0.0001	5	viterbi	0.1	42	0.1642	0.471	0.465	0.164	0.4647316	0.9142	0.828	0.1014	0.9	0.1719

Appendix GG: All results in Auto-Encoder Model for Experiment 7.

Threshold Method																		
Tuning Parameters									**Evaluations**									
nb_epoch	**batch_size**	**input_dim**	**encoding_dim**	**hidden_dim1**	**hidden_dim2**	**activation**	**learning_rate**	**Threshold**	**Accuracy**	**Precision**	**Recall**	**F1-score**	**ROC auc score**	**RMSE**	**TPR**	**TNR**	**FPR**	**FNR**
10	128	58	18	10	6	tanh	1.00E-07	4	0.9077	0.5235	0.5024	0.4871	0.5024	0.3038	0.0124	0.9924	0.0076	0.9876
50	128	58	18	10	6	tanh	1.00E-07	4	0.906	0.5233	0.503	0.4895	0.503	0.3066	0.0159	0.9901	0.0099	0.9841
10	128	58	32	16	8	tanh	1.00E-07	4	0.907	0.5241	0.5027	0.4883	0.5027	0.305	0.0141	0.9914	0.0086	0.9859
10	128	58	10	5	2	tanh	1.00E-07	4	0.9037	0.521	0.5035	0.4917	0.5035	0.3103	0.0196	0.9873	0.0127	0.9804
10	128	58	5	2	1	tanh	1.00E-07	4	0.903	0.51997	0.5035	0.4921	0.5035	0.3115	0.0206	0.9864	0.0136	0.9794
10	128	58	5	3	1	tanh	1.00E-07	4	0.9035	0.523	0.5039	0.4927	0.5039	0.3106	0.0209	0.987	0.013	0.9791

10	128	58	50	20	10	tanh	1.00E-07	4	0.9086	0.5226	0.5019	0.4856	0.5019	0.3023	0.0103	0.9935	0.0065	0.9897
10	12	58	50	20	10	tanh	1.00E-07	4	0.9084	0.525	0.5023	0.4864	0.5023	0.3027	0.0113	0.9932	0.0068	0.9887
10	12	58	5	2	1	tanh	1.00E-07	4	0.9031	0.5207	0.5037	0.4924	0.5037	0.3114	0.0208	0.9865	0.0135	0.9792
10	256	58	5	2	1	tanh	1.00E-07	4	0.903	0.5212	0.5038	0.4927	0.5038	0.3115	0.0212	0.9864	0.0136	0.9788
10	128	58	5	2	1	sigmoid	1.00E-07	4	0.9028	0.5213	0.5039	0.4929	0.5039	0.3117	0.0216	0.9861	0.0139	0.9784
10	128	58	5	2	1	hard_sigmoid	1.00E-07	4	0.9018	0.5223	0.5045	0.4945	0.5045	0.3134	0.0242	0.9848	0.0152	0.9758
10	128	58	5	2	1	exponential	1.00E-07	4	0.9018	0.5223	0.5045	0.4945	0.5045	0.3134	0.0242	0.9848	0.0152	0.9758
10	128	58	5	2	1	linear	1.00E-07	4	0.9031	0.5184	0.5032	0.4915	0.5032	0.3113	0.0197	0.9867	0.0133	0.9803
10	128	58	5	2	1	tanh	1.00E-07	3	0.8966	0.5173	0.5048	0.4974	0.5048	0.3215	0.0311	0.9785	0.0215	0.9689
10	128	58	5	2	1	tanh	1.00E-07	2	0.8788	0.5285	0.5167	0.5181	0.5167	0.3482	0.0789	0.9544	0.0456	0.9211

10	128	58	5	2	1	tanh	1.00E-07	1	0.4668	0.5133	0.5415	0.3886	0.5415	0.7302	0.6318	0.4512	0.5488	0.3682
10	128	58	5	2	1	tanh	1.00E-07	5	0.9085	0.5313	0.5029	0.4876	0.5029	0.3025	0.0125	0.9932	0.0068	0.9875
10	128	58	5	2	1	linear	1.00E-06	4	0.9032	0.5186	0.5032	0.4915	0.5032	0.3112	0.0197	0.9867	0.0133	0.9803
10	128	58	5	2	1	tanh	1.00E-08	4	0.9029	0.521	0.5038	0.4926	0.5038	0.3115	0.0212	0.9863	0.0137	0.9788
10	128	58	5	2	1	tanh	1.00E-09	4	0.9029	0.521	0.5038	0.4926	0.5038	0.3115	0.0212	0.9863	0.0137	0.9788
10	128	58	5	2	1	tanh	1.00E-06	4	0.903	0.52111	0.5038	0.4926	0.5038	0.3114	0.0211	0.9864	0.0136	0.9789

Appendix HH: All results in K-means Model for Experiment 8.

two clusters method													
Tuning Parameters					**Evaluations**								
initialization	**n_init**	**max_iter**	**algorithm**	**RandomState**	**Accuracy**	**Precision**	**Recall**	**F1-score**	**ROC auc score**	**RMSE**	**TPR**	**TNR**	**FPR**	**FNR**
K-means++	5	1	auto	0	0.5152	0.5022	0.5032	0.4656	0.5032	0.6963	0.4819	0.5245	0.4755	0.5181

K-means++	5	10	auto	0	0.4982	0.5011	0.5016	0.4567	0.5016	0.7084	0.5077	0.4956	0.5044	0.4923
K-means++	5	1	auto	42	0.4796	0.4748	0.4631	0.4318	0.4631	0.7214	0.4337	0.4925	0.5075	0.5663
K-means++	5	10	auto	42	0.5008	0.5087	0.5127	0.4623	0.5127	0.7065	0.5339	0.4916	0.5084	0.4661
K-means++	5	1	auto	1	0.553	0.4856	0.4801	0.4679	0.4801	0.6686	0.3508	0.6094	0.3906	0.6492
K-means++	5	10	auto	1	0.527	0.5489	0.5712	0.4972	0.5712	0.6878	0.6496	0.4927	0.5073	0.3504
K-means++	5	1	auto	2	0.5003	0.4899	0.4852	0.4508	0.4852	0.7069	0.4584	0.512	0.488	0.5416
K-means++	5	10	auto	2	0.4915	0.486	0.4795	0.4443	0.4795	0.7131	0.4582	0.5008	0.4992	0.5418
K-means++	5	1	auto	3	0.4843	0.4894	0.4845	0.4429	0.4845	0.7181	0.4849	0.4841	0.5159	0.5151
K-means++	5	10	auto	3	0.4766	0.4588	0.4399	0.4198	0.4399	0.7235	0.3748	0.5051	0.4949	0.6252
K-means++	5	1	auto	4	0.5009	0.4809	0.4722	0.4451	0.4722	0.7065	0.4212	0.5231	0.4769	0.5788
K-means++	5	10	auto	4	0.5029	0.5055	0.5081	0.4616	0.5081	0.705	0.5172	0.4989	0.5011	0.4828
K-means++	5	1	auto	5	0.4781	0.5123	0.5177	0.4512	0.5177	0.7224	0.588	0.4474	0.5526	0.412
K-means++	5	10	auto	5	0.505	0.5124	0.5181	0.4665	0.5181	0.7036	0.5414	0.4948	0.5052	0.4586
K-means++	5	1	auto	13	0.4599	0.5067	0.5095	0.4377	0.5095	0.7349	0.5976	0.4215	0.5785	0.4024
K-means++	5	10	auto	13	0.5184	0.5234	0.5343	0.4797	0.5343	0.694	0.5624	0.5061	0.4939	0.4376

K-means++	5	1	auto	14	0.4479	0.4909	0.487	0.4237	0.487	0.743	0.5565	0.4176	0.5824	0.4435
K-means++	5	10	auto	14	0.502	0.5074	0.5108	0.4622	0.5108	0.7057	0.5263	0.4952	0.5048	0.4737
K-means++	5	1	auto	90	0.5183	0.4975	0.4963	0.4639	0.4963	0.694	0.4573	0.5354	0.4646	0.5427
K-means++	5	10	auto	90	0.5037	0.5047	0.5069	0.4615	0.5069	0.7045	0.5125	0.5013	0.4987	0.4875
K-means++	5	1	auto	91	0.4208	0.467	0.4531	0.3972	0.4531	0.761	0.5103	0.3959	0.6041	0.4897
K-means++	5	10	auto	91	0.4818	0.4766	0.4658	0.4339	0.4658	0.7199	0.4375	0.4941	0.5059	0.5625
K-means++	5	1	auto	92	0.5445	0.525	0.5364	0.4938	0.5364	0.6749	0.522	0.5508	0.4492	0.478
K-means++	5	10	auto	92	0.4841	0.481	0.4721	0.4377	0.4721	0.7183	0.4509	0.4933	0.5067	0.5491
K-means++	5	1	auto	200	0.5336	0.4912	0.4875	0.4657	0.4875	0.6829	0.4055	0.5694	0.4306	0.5945
K-means++	5	10	auto	200	0.4905	0.4839	0.4764	0.4425	0.4764	0.7138	0.4512	0.5015	0.4985	0.5488
K-means++	5	1	auto	250	0.5212	0.5144	0.5211	0.476	0.5211	0.6919	0.5207	0.5214	0.4786	0.4793
K-means++	5	10	auto	250	0.5083	0.5111	0.5162	0.4675	0.5162	0.7012	0.5303	0.5021	0.4979	0.4697
random	5	1	Full	5	0.5037	0.5047	0.5069	0.4615	0.5069	0.7045	0.5125	0.5013	0.4987	0.4875
random	5	10	elkan	5	0.4208	0.467	0.4531	0.3972	0.4531	0.761	0.5103	0.3959	0.6041	0.4897
random	10	10	auto	5	0.4818	0.4766	0.4658	0.4339	0.4658	0.7199	0.4375	0.4941	0.5059	0.5625

Appendix II: All results in HMM Model for Experiment 8.

two states method															
Tunimg Parameters						**Evaluations**									
covariance_type	**min_covar**	**n_iter**	**algorithm**	**tol**	**Random State**	**Accuracy**	**Precision**	**Recall**	**F1-score**	**ROC auc score**	**RMSE**	**TPR**	**TNR**	**FPR**	**FNR**
spherical	0.0001	5000	viterbi	0.1	defaults	0.5077	0.4915	0.488	0.455	0.4875385	0.7017	0.452	0.5233	0.48	0.5482
diag	0.0001	5000	viterbi	0.1	defaults	0.7519	0.5515	0.522	0.511	0.5221504	0.4981	0.114	0.93	0.07	0.8857
tied	0.0001	5000	viterbi	0.1	defaults	0.5317	0.5444	0.565	0.498	0.5648735	0.6843	0.624	0.506	0.49	0.3763
full	0.0001	defaults	viterbi	0.1	defaults	0.5251	0.5442	0.564	0.494	0.56445	0.6891	0.634	0.4946	0.51	0.3657
spherical	0.0001	defaults	viterbi	defaults	defaults	0.4955	0.4861	0.48	0.446	0.4796708	0.7103	0.452	0.5078	0.49	0.5484
diag	0.0001	defaults	viterbi	defaults	defaults	0.4991	0.4958	0.494	0.454	0.4939049	0.7078	0.485	0.5031	0.5	0.5153
tied	0.0001	defaults	viterbi	defaults	defaults	0.4948	0.4943	0.492	0.451	0.4915896	0.7107	0.486	0.4974	0.5	0.5142
full	0.0001	defaults	viterbi	defaults	defaults	0.4755	0.4603	0.442	0.42	0.442063	0.7242	0.383	0.5015	0.5	0.6174
spherical	0.0001	5000	map	0.1	defaults	0.4892	0.4909	0.487	0.446	0.4866855	0.7147	0.482	0.4911	0.51	0.5178
diag	0.0001	5000	map	0.1	defaults	0.4309	0.3966	0.351	0.356	0.3506077	0.7544	0.208	0.4931	0.51	0.7919
tied	0.0001	5000	map	0.1	defaults	0.5139	0.5106	0.515	0.47	0.5154891	0.6972	0.518	0.5126	0.49	0.4816
full	0.0001	defaults	map	0.1	defaults	0.4922	0.4951	0.493	0.45	0.4928266	0.7126	0.494	0.4918	0.51	0.5061

spherical	0.0001	defaults	map	defaults	defaults	0.4991	0.4972	0.496	0.455	0.4959548	0.7077	0.49	0.5016	0.5	0.5097
diag	0.0001	defaults	map	defaults	defaults	0.47	0.4503	0.428	0.411	0.4275683	0.728	0.352	0.5029	0.5	0.6477
tied	0.0001	defaults	map	defaults	defaults	0.4908	0.4876	0.482	0.445	0.4817751	0.7136	0.466	0.4978	0.5	0.5343
full	0.0001	defaults	map	defaults	defaults	0.5501	0.5746	0.608	0.523	0.6084653	0.6707	0.712	0.5049	0.5	0.288
spherical	0.0001	5000	viterbi	defaults	defaults	0.4955	0.4861	0.48	0.446	0.4796708	0.7103	0.452	0.5078	0.49	0.5484
spherical	0.0001	5	viterbi	0.1	defaults	0.4959	0.4962	0.494	0.453	0.4944799	0.71	0.492	0.497	0.5	0.508
spherical	0.0001	5	viterbi	0.1	42	0.4964	0.4928	0.489	0.451	0.4894425	0.7097	0.477	0.5017	0.5	0.5228

Appendix JJ: All results in Auto-Encoder Model for Experiment 8.

Threshold Method																		
Tuning Parameters									**Evaluations**									
nb_epoch	**batch_size**	**input_dim**	**encoding_dim**	**hidden_dim1**	**hidden_dim2**	**activation**	**learning_rate**	**Threshold**	**Accuracy**	**Precision**	**Recall**	**F1-score**	**ROC auc score**	**RMSE**	**TPR**	**TNR**	**FPR**	**FNR**
10	128	201	18	10	6	tanh	1.00E-07	4	0.7817	0.3908	0.5	0.4387	0.5	0.4672	0	1	0	1

50	128	201	18	10	6	tanh	1.00E-07	4	0.7817	0.3908	0.5	0.4387	0.5	0.4672	0	1	0	1
10	128	201	32	16	8	tanh	1.00E-07	4	0.7817	0.3908	0.5	0.4387	0.5	0.4672	0	1	0	1
10	128	201	10	5	2	tanh	1.00E-07	4	0.7817	0.3908	0.5	0.4387	0.5	0.4672	0	1	0	1
10	128	201	5	2	1	tanh	1.00E-07	4	0.7817	0.3908	0.5	0.4387	0.5	0.4672	0	1	0	1
10	128	201	5	3	1	tanh	1.00E-07	4	0.7817	0.3908	0.5	0.4387	0.5	0.4672	0	1	0	1
10	128	201	50	20	10	tanh	1.00E-07	4	0.7817	0.3908	0.5	0.4387	0.5	0.4672	0	1	0	1
10	12	201	50	20	10	tanh	1.00E-07	4	0.7817	0.3908	0.5	0.4387	0.5	0.4672	0	1	0	1
10	12	201	5	2	1	tanh	1.00E-07	4	0.7817	0.3908	0.5	0.4387	0.5	0.4672	0	1	0	1
10	256	201	5	2	1	tanh	1.00E-07	4	0.7817	0.3908	0.5	0.4387	0.5	0.4672	0	1	0	1
10	128	201	5	2	1	sigmoid	1.00E-07	4	0.7817	0.3908	0.5	0.4387	0.5	0.4672	0	1	0	1

10	128	201	5	2	1	hard_sigmoid	1.00E-07	4	0.7817	0.3908	0.5	0.4387	0.5	0.4672	0	1	0	1
10	128	201	5	2	1	exponential	1.00E-07	4	0.7817	0.3908	0.5	0.4387	0.5	0.4672	0	1	0	1
10	128	201	5	2	1	linear	1.00E-07	4	0.7817	0.3908	0.5	0.4387	0.5	0.4672	0	1	0	1
10	128	201	5	2	1	tanh	1.00E-07	3	0.7817	0.3908	0.5	0.4387	0.5	0.4672	0	1	0	1
10	128	201	5	2	1	tanh	1.00E-07	2	0.7817	0.8908	0.5	0.4388	0.5	0.4672	9.95E-05	1	0	0.9999
10	128	201	5	2	1	tanh	1.00E-07	1	0.5721	0.5635	0.5929	0.5307	0.5929	0.6541	0.6297114	0.556	0.444	0.3703
10	128	201	5	2	1	tanh	1.00E-07	5	0.7817	0.3908	0.5	0.4387	0.5	0.4672	0	1	0	1
10	128	201	5	2	1	linear	1.00E-06	4	0.7817	0.3908	0.5	0.4387	0.5	0.4672	0	1	0	1
10	128	201	5	2	1	tanh	1.00E-08	4	0.7817	0.3908	0.5	0.4387	0.5	0.4672	0	1	0	1
10	128	201	5	2	1	tanh	1.00E-09	4	0.7817	0.3908	0.5	0.4387	0.5	0.4672	0	1	0	1

10	128	201	5	2	1	tanh	1.00E-06	4	0.7817	0.3908	0.5	0.4387	0.5	0.4672	0	1	0	1

Appendix KK: All results in K-means Model for Experiment 9.

two clusters method														
Tuning Parameters					**Evaluations**									
initialization	**n_init**	**max_iter**	**algorithm**	**RandomState**	**Accuracy**	**Precision**	**Recall**	**F1-score**	**ROC auc score**	**RMSE**	**TPR**	**TNR**	**FPR**	**FNR**
K-means++	5	1	auto	0	0.4156	0.4197	0.4259	0.409	0.4259	0.7645	0.5632	0.2886	0.7114	0.4368
K-means++	5	10	auto	0	0.5169	0.5114	0.5112	0.5105	0.5112	0.6951	0.4361	0.5864	0.4136	0.5639
K-means++	5	1	auto	42	0.5083	0.5038	0.5037	0.5034	0.5037	0.7012	0.4424	0.5651	0.4349	0.5576
K-means++	5	10	auto	42	0.5169	0.5114	0.5112	0.5105	0.5112	0.6951	0.4361	0.5864	0.4136	0.5639
K-means++	5	1	auto	1	0.4896	0.4884	0.4883	0.4881	0.4883	0.7144	0.4714	0.5052	0.4948	0.5286
K-means++	5	10	auto	1	0.4831	0.4886	0.4888	0.4824	0.4888	0.7189	0.5639	0.4136	0.5864	0.4361
K-means++	5	1	auto	2	0.5184	0.5369	0.5322	0.5082	0.5322	0.694	0.7161	0.3483	0.6517	0.2839
K-means++	5	10	auto	2	0.4831	0.4886	0.4888	0.4824	0.4888	0.7189	0.5639	0.4136	0.5864	0.4361
K-means++	5	1	auto	3	0.4627	0.4805	0.4941	0.3718	0.4941	0.733	0.9116	0.0766	0.9234	0.0884

K-means++	5	10	auto	3	0.5117	0.5163	0.5161	0.5116	0.5161	0.6988	0.5746	0.4577	0.5423	0.4254
K-means++	5	1	auto	4	0.5018	0.5209	0.5174	0.4873	0.5174	0.7058	0.7247	0.3101	0.6899	0.2753
K-means++	5	10	auto	4	0.4831	0.4886	0.4888	0.4824	0.4888	0.7189	0.5639	0.4136	0.5864	0.4361
K-means++	5	1	auto	5	0.5217	0.5209	0.521	0.5206	0.521	0.6916	0.5118	0.5302	0.4698	0.4882
K-means++	5	10	auto	5	0.4831	0.4886	0.4888	0.4824	0.4888	0.7189	0.5639	0.4136	0.5864	0.4361
K-means++	5	1	auto	13	0.4844	0.4773	0.4779	0.4765	0.4779	0.718	0.3907	0.5651	0.4349	0.6093
K-means++	5	10	auto	13	0.5169	0.5114	0.5112	0.5105	0.5112	0.6951	0.4361	0.5864	0.4136	0.5639
K-means++	5	1	auto	14	0.4599	0.4588	0.4953	0.3353	0.4953	0.7349	0.9655	0.025	0.975	0.0345
K-means++	5	10	auto	14	0.4839	0.4894	0.4896	0.4831	0.4896	0.7184	0.5663	0.413	0.587	0.4337
K-means++	5	1	auto	90	0.593	0.5897	0.5888	0.5889	0.5888	0.638	0.5333	0.6443	0.3557	0.4667
K-means++	5	10	auto	90	0.4831	0.4886	0.4888	0.4824	0.4888	0.7189	0.5639	0.4136	0.5864	0.4361
K-means++	5	1	auto	91	0.4897	0.4903	0.4902	0.4893	0.4902	0.7143	0.4971	0.4834	0.5166	0.5029
K-means++	5	10	auto	91	0.5169	0.5114	0.5112	0.5105	0.5112	0.6951	0.4361	0.5864	0.4136	0.5639
K-means++	5	1	auto	92	0.5377	0.5359	0.5361	0.5359	0.5361	0.6799	0.5139	0.5582	0.4418	0.4861
K-means++	5	10	auto	92	0.4831	0.4886	0.4888	0.4824	0.4888	0.7189	0.5639	0.4136	0.5864	0.4361

K-means++	5	1	auto	200	0.5664	0.5663	0.5667	0.5658	0.5667	0.6585	0.5695	0.5638	0.4362	0.4305
K-means++	5	10	auto	200	0.5169	0.5114	0.5112	0.5105	0.5112	0.6951	0.4361	0.5864	0.4136	0.5639
K-means++	5	1	auto	250	0.5431	0.5342	0.5304	0.5223	0.5304	0.676	0.362	0.6988	0.3012	0.638
K-means++	5	10	auto	250	0.4831	0.4886	0.4888	0.4824	0.4888	0.7189	0.5639	0.4136	0.5864	0.4361
random	5	1	Full	5	0.5083	0.5038	0.5037	0.5034	0.5037	0.7012	0.4424	0.5651	0.4349	0.5576
random	5	10	elkan	5	0.5169	0.5114	0.5112	0.5105	0.5112	0.6951	0.4361	0.5864	0.4136	0.5639
random	10	10	auto	5	0.4896	0.4884	0.4883	0.4881	0.4883	0.7144	0.4714	0.5052	0.4948	0.5286

Appendix LL: All results in HMM Model for Experiment 9.

two states method														
Tuning Parameters						**Evaluations**								
covariance_type	**min_covar**	**n_iter**	**algorithm**	**tol**	**Random State**	**Accuracy**	**Precision**	**Recall**	**F1-score**	**ROC auc score**	**RMSE**	**TPR**	**TNR**	**FPR**	**FNR**
spherical	0.0001	5000	viterbi	0.1	defaults	0.6267	0.6326	0.632	0.627	0.6316545	0.611	0.698	0.5654	0.43	0.3021
diag	0.0001	5000	viterbi	0.1	defaults	0.3923	0.3904	0.39	0.39	0.3900222	0.7795	0.36	0.4204	0.58	0.6404
tied	0.0001	5000	viterbi	0.1	defaults	0.5182	0.5129	0.513	0.512	0.5127322	0.6941	0.44	0.5853	0.41	0.5599
full	0.0001	5000	viterbi	0.1	defaults	0.8593	0.8833	0.869	0.859	0.8691058	0.3751	1	0.7383	0.26	0.0001

spherical	0.0001	defaults	viterbi	defaults	defaults	0.6267	0.6327	0.632	0.627	0.6317295	0.611	0.698	0.5653	0.43	0.3018
diag	0.0001	defaults	viterbi	defaults	defaults	0.6077	0.6096	0.61	0.608	0.6099778	0.6263	0.64	0.5796	0.42	0.3596
tied	0.0001	defaults	viterbi	defaults	defaults	0.4818	0.4871	0.487	0.481	0.4872678	0.7199	0.56	0.4147	0.59	0.4401
full	0.0001	defaults	viterbi	defaults	defaults	0.8593	0.8833	0.869	0.859	0.8691058	0.3751	1	0.7383	0.26	0.0001
spherical	0.0001	5000	map	0.1	defaults	0.3733	0.3674	0.368	0.368	0.3683455	0.7916	0.302	0.4346	0.57	0.6979
diag	0.0001	5000	map	0.1	defaults	0.3923	0.3904	0.39	0.39	0.3900222	0.7795	0.36	0.4204	0.58	0.6404
tied	0.0001	5000	map	0.1	defaults	0.4818	0.4871	0.487	0.481	0.4872678	0.7199	0.56	0.4147	0.59	0.4401
full	0.0001	5000	map	0.1	defaults	0.8593	0.8833	0.869	0.859	0.8691058	0.3751	1	0.7383	0.26	0.0001
spherical	0.0001	defaults	map	defaults	defaults	0.3733	0.3673	0.368	0.368	0.3682705	0.7917	0.302	0.4347	0.57	0.6982
diag	0.0001	defaults	map	defaults	defaults	0.6077	0.6096	0.61	0.608	0.6099778	0.6263	0.64	0.5796	0.42	0.3596
tied	0.0001	defaults	map	defaults	defaults	0.4818	0.4871	0.487	0.481	0.4872678	0.7199	0.56	0.4147	0.59	0.4401
full	0.0001	defaults	map	defaults	defaults	0.8593	0.8833	0.869	0.859	0.8691058	0.3751	1	0.7383	0.26	0.0001
spherical	0.0001	5000	viterbi	defaults	defaults	0.6267	0.6326	0.632	0.627	0.6316545	0.611	0.698	0.5654	0.43	0.3021
spherical	0.0001	5	viterbi	0.1	defaults	0.3731	0.3668	0.368	0.367	0.3679181	0.7917	0.298	0.4374	0.56	0.7016
spherical	0.0001	5	viterbi	0.1	42	0.6269	0.6332	0.632	0.627	0.6320819	0.6109	0.702	0.5626	0.44	0.2984

Appendix MM: All results in Auto-Encoder Model for Experiment 9.

Threshold Method																		
Tuning Parameters									**Evaluations**									
nb_epoch	**batch_size**	**input_dim**	**encoding_dim**	**hidden_dim1**	**hidden_dim2**	**activation**	**learning_rate**	**Threshold**	**Accuracy**	**Precision**	**Recall**	**F1-score**	**ROC auc score**	**RMSE**	**TPR**	**TNR**	**FPR**	**FNR**
10	128	128	18	10	6	tanh	1.00E-07	4	0.553	0.6434	0.5179	0.3995	0.5179	0.6686	0.0513	0.9845	0.0155	0.9487
50	128	128	18	10	6	tanh	1.00E-07	4	0.5524	0.6462	0.5172	0.3973	0.5172	0.669	0.0487	0.9857	0.0143	0.9513
10	128	128	32	16	8	tanh	1.00E-07	4	0.5525	0.6455	0.5173	0.3976	0.5173	0.6689	0.0491	0.9855	0.0145	0.9509
10	128	128	10	5	2	tanh	1.00E-07	4	0.5548	0.6476	0.5199	0.404	0.5199	0.6673	0.0561	0.9837	0.0163	0.9439
10	128	128	5	2	1	tanh	1.00E-07	4	0.5558	0.6462	0.5211	0.4075	0.5211	0.6665	0.06	0.9822	0.0178	0.94
10	128	128	5	3	1	tanh	1.00E-07	4	0.5556	0.6483	0.5209	0.4064	0.5209	0.6666	0.0587	0.983	0.017	0.9413

10	128	128	50	20	10	tanh	1.00E-07	4	0.55505	0.6525	0.5148	0.3901	0.5148	0.6704	0.0408	0.9889	0.0111	0.9592
10	12	128	50	20	10	tanh	1.00E-07	4	0.55506	0.6631	0.5148	0.3887	0.5148	0.6704	0.0389	0.9906	0.0094	0.9611
10	12	128	5	2	1	tanh	1.00E-07	4	0.55548	0.643	0.52	0.4051	0.52	0.6673	0.0575	0.9825	0.0175	0.9425
10	256	128	5	2	1	tanh	1.00E-07	4	0.55552	0.6441	0.5205	0.4063	0.5205	0.6669	0.0588	0.9822	0.0178	0.9412
10	128	128	5	2	1	sigmoid	1.00E-07	4	0.55551	0.6392	0.5204	0.407	0.5204	0.667	0.0599	0.981	0.019	0.9401
10	128	128	5	2	1	hard_sigmoid	1.00E-07	4	0.55552	0.6407	0.5206	0.4072	0.5206	0.6669	0.06	0.9812	0.0188	0.94
10	128	128	5	2	1	exponential	1.00E-07	4	0.55552	0.6407	0.5206	0.4072	0.5206	0.6669	0.06	0.9812	0.0188	0.94
10	128	128	5	2	1	linear	1.00E-07	4	0.55541	0.6426	0.5193	0.4033	0.5193	0.6677	0.0555	0.983	0.017	0.9445
10	128	128	5	2	1	tanh	1.00E-07	3	0.57225	0.6308	0.5416	0.4633	0.5416	0.6539	0.1313	0.9519	0.0481	0.8687
10	128	128	5	2	1	tanh	1.00E-07	2	0.6049	0.626	0.5834	0.5566	0.5834	0.6286	0.2971	0.8696	0.1304	0.7029

10	128	128	5	2	1	tanh	1.00E-07	1	0.6278	0.6257	0.6257	0.6257	0.6257	0.6101	0.5972	0.6541	0.3459	0.4028
10	128	128	5	2	1	tanh	1.00E-07	5	0.5445	0.6357	0.5087	0.3751	0.5087	0.6745	0.0255	0.9919	0.0081	0.9745
10	128	128	5	2	1	linear	1.00E-06	4	0.5542	0.6417	0.5194	0.4037	0.5194	0.6677	0.0561	0.9827	0.0173	0.9439
10	128	128	5	2	1	tanh	1.00E-08	4	0.5554	0.6446	0.5207	0.4066	0.5207	0.6668	0.0591	0.9822	0.0178	0.9409
10	128	128	5	2	1	tanh	1.00E-09	4	0.5548	0.6445	0.52	0.4049	0.52	0.6672	0.0572	0.9828	0.0172	0.9428
10	128	128	5	2	1	tanh	1.00E-06	4	0.5554	0.6446	0.5207	0.4066	0.5207	0.6668	0.0591	0.9822	0.0178	0.9409

Curriculum Vitae

Name: Iman Abu Sulayman

Post-secondary Education and Degrees:
TAIF University
Taif, Makkah, Saudi Arabia
2010-2014 B.A.

Related Work Experience
Teaching Assistant
The University of Western Ontario
2018-2019

Teaching Assistant
Taif University
2015-2016

Publications:

- **Iman I. M. Abu Sulayman** and Abdelkader Ouda, "Human Trait Analysis via Machine Learning Techniques for User Authentication", Submitted to the First IEEE International Conference on Trust, Privacy and Security in Intelligent Systems, and Applications, Los Angeles, California, USA 2019.
- **Iman I. M. Abu Sulayman** and Abdelkader Ouda, "User Modeling via Anomaly Detection Techniques for User Authentication", The 10th Annual Information Technology, Electronics and Mobile Communication Conference IEEE IEMCON 2019. University of British Columbia, Vancouver, Canada, 2019.
- **Iman I. M. Abu Sulayman** and Abdelkader Ouda, "Data Analytics Methods for Anomaly Detection: Evolution and Recommendations", International Conference on Signal Processing and Information Security (Icspis 2018), University of Dubai, Academic City, 2018.
- Mohamed S. Soliman, Majed O. Dwairi, and **Iman I. M. Abu Sulayman,** "The Effect of the Ground Slots up on the Bandwidth Performance for UWB Antenna ", Accepted at the 18th Mediterranean Microwave Symposium MMS, Istanbul, 2018.

- Mohamed S. Soliman, Majed O. Dwairi, **Iman I. M. Abu Sulayman,** Sami H. A. Almalki, "Design and Performance Analysis of Fractal Regular Slotted-Patch Antennas for Ultra-Wideband Communication Systems", IET Microwaves, Antennas & Propagation, 2017.
- M. O. Dwairi, M. S. Soliman, A. A. Alahmadi, **I. I. M. A. Sulayman** and S. H. A. Almalki, "Design regular fractal slot-antennas for ultra-wideband applications," 2017 Progress In Electromagnetics Research Symposium - Spring (PIERS), St. Petersburg, 2017, pp. 3875-3880. doi: 10.1109/PIERS.2017.8262435
- **Iman I. M. Abu Sulayman**, Sami H. A. Almalki, Mohamed S. Soliman, Majed O. Dwairi,, "Designing and Implementation of Home Automation System Based on Remote Sensing Technique with Arduino Uno Microcontroller", 9th IEEE-GCC, 2017.
- Sami H. A. Almalki, **Iman I. M. Abu Sulayman**, Mohamed S. Soliman, Majed O. Dwairi,, " Designing Reliable Dual Mode RealTime Home Automation System Based on Very High Speed Description Language", 9th IEEE-GCC, 2017.
- S. H. A. Almalki, **I. I. M. Abu Sulayman**, M. O. Dwairi and M. S. Soliman, "Designing Reliable Dual Mode Real-Time Home Automation System Based on Very High Speed Description Language," *2017 9th IEEE-GCC Conference and Exhibition (GCCCE)*, Manama, Bahrain, 2017, pp. 1-4. doi: 10.1109/IEEEGCC.2017.8448166
- **Iman I. M. Abosolaiman**, Sami H. A. Almalki, Mohamed S. Soliman, "Designing Reliable Dual Mode Real-Time Home Automation System Based on Very High Speed Description Language", International Journal of Control, Automation and Systems, Vol.5, No.3, July 2016.
- Mohamed S. Soliman, Majed O. Dwairi, **Iman I. M. Abu Sulayman**, Sami H. A. Almalki, "a comparative study for designing and modelingpatch antenna with different electromagnetic cadapproaches – a case study", International Journal on Communications Antenna and Propagation (IRECAP),Vol.6, No.2, 2016.

- **Iman I. M. Abu Sulayman**, Sami H. A. Almalki, Mohamed S. Soliman, Majed O. Dwairi, “Design and Performance Analysis of Patch Antenna for Microwave Radio-Frequency Energy Harvesting System”, International Conference on Electromagnetic in Advanced Applications, April, 2016.
- **I. M. A. Sulayman**, S. H. A. Almalki, M. S. Soliman and M. O. Dwairi, "A comparative study for designing and modeling patch antenna with different electromagnetic CAD approaches — A case study," *2016 Progress in Electromagnetic Research Symposium (PIERS)*, Shanghai, 2016, pp. 2803-2806. doi: 10.1109/PIERS.2016.7735128
- **Iman I. M. Abu Sulayman**, Sami H. A. Almalki, Mohamed S. Soliman, “Design and Implementation of a Reliable Wireless Realtime Home Automation System Based on Arduino Uno Single-board Microcontroller”, Progress in Electromagnetics Research Symposium (PIERS), July, 2015.
- **Iman I. M. Abosolaiman**, Sami H. A. Almalki, Mohamed S. Soliman, Nadjim Merabtine, " Designing and Implementation of Home Automation System Based on Remote Sensing Technique with Arduino Uno Microcontroller ", Sixth Science Conference for Higher Education Students, December 2014.
- Bader M. O. Al-thobaiti**, Iman I. M. Abosolaiman**, Mahdi H. M. Alzahrani, Sami H. A. Almalki, Mohamed S. Soliman, “Design and Implementation of a Reliable Wireless Real-Time Home Automation System Based on Arduino Uno Single-Board Microcontroller”, International Journal of Control, Automation and Systems, Vol.3, No.3, pp .11-15, July 2014.
- Bader M. O. Al-thobaiti**, Iman I. M. Abosolaiman**, Mahdi H. M. Alzahrani, Sami H. A. Almalki, Mohamed S. Soliman, “Designing a Reliable Wireless Dual Mode Real-time Home Automation System Based on Arduino Single-board Microcontroller”, Progress in Electromagnetics Research Symposium (PIERS), April 2014.
- Bader M. O. Al-thobaiti**, Iman I. M. Abosolaiman**, Mahdi H. M. Alzahrani, Sami H. A. Almalki, Mohamed S. Soliman, " Designing a

Reliable Dual Mode Real-Time Home Automation System Based on Field Programmable Gate Array Controller ", Fourth Science Conference for Higher Education Students, December 2012.