

# A New Feature Engineering Framework for Financial Cyber Fraud Detection Using Machine Learning and Deep Learning

This dissertation is submitted for the degree of  
Doctor of Philosophy

**BY**

**Chie Ikeda**

Date: September 2022



**School of Computing and Digital Media**

London Metropolitan University

166-220 Holloway Road, London, N7 8DB

## **Acknowledgements**

I would like to acknowledge and give my warmest thanks to many people, without them it would not have been possible to complete my journey. I would first like to express my gratitude to my supervisor, Prof. Karim Ouazzane who made this work possible and provided extraordinary support, guidance and advice carried me through all the stages of my research. Always motivated me when things got hard, always pointed me in the right direction and encouraged me to never give up. I would further like to thank my second advisor, Dr. Qicheng Yu, who have also motivated me and supported me with his exact advice and feedback. This project would not have been as good without his help.

Finally, I would also like to thank my family, especially my husband for understanding my condition and being extremely supportive.

Thank you very much, everyone.

Chie Ikeda

I would like to dedicate this thesis to my future self...

## **Abstract:**

As online payment system advances, the total losses via online banking in the United Kingdom have increased because fraudulent techniques have also progressed and used advanced technology. Using traditional fraud detection models with only raw transaction data cannot cope with the emerging new and innovative scheme to deceive financial institutions. Many studies published by both academic and commercial organisations introduce new fraud detection models using various machine learning algorithms, however, financial fraud losses via the online banking have been still increasing. This thesis looks at the holistic views of feature engineering for classification and machine learning (ML) and deep learning (DL) algorithms for fraud detection to understand their capabilities and how to deal with input data in each algorithm. And then, proposes a new feature engineering framework that can produce the most effective features set for any ML and DL algorithms by taking both methods of feature engineering and features selection into a new framework. The framework consists of two main components: feature creation and feature selection. The purpose of feature creation component is to create many effective feature candidates by feature aggregation and transformation based on customer's behaviour. The purpose of feature selection is to evaluate all features and to drop irrelevant features and very high correlated features from the dataset. In the experiment, I proved the effect of using a new feature engineering framework by using a real-life banking transactional data provided by a private European bank and evaluating performances of the built fraud detection models in an appropriate way. Machine Learning and Deep learning models perform at their best when the created features set by the new framework are applied with higher scores in all evaluation metrics compared to the scores of the models built with the original dataset.

## List of Publications

- **Chie Ikeda, Karim Ouazzane, Qicheng Yu.** New Feature Engineering Framework for Machine Learning in Financial Fraud Detection. Conference: 10<sup>th</sup> International Conference on Advances in Computing and Information Technology (ACITY 2020). November 2020.
- **Chie Ikeda, Karim Ouazzane, Qicheng Yu, Svetla Hubenova.** New Feature Engineering Framework for Deep Learning in Financial Fraud Detection. International Journal of Advanced Computer Science and Applications (IJACSA), December 2021. Volume: 12 Issue 12.

## Contents

<b>1. Introduction</b> .....	16
<b>1.1. Research Problem</b> .....	22
<b>1.2. Aim and Objectives of this research</b> .....	23
<b>1.3. Research Methodology</b> .....	26
<b>1.3.1. Analytical Approach</b> .....	27
<b>1.3.2. Practical Approach</b> .....	29
<b>1.3.3. Observation Approach</b> .....	30
<b>1.4. Summary Overview Diagram</b> .....	31
<b>1.5. The Contributions to Knowledge</b> .....	32
<b>1.6. Structure of the thesis</b> .....	33
<b>2. Literature Review</b> .....	35
<b>2.1. Machine Learning Algorithms</b> .....	35
<b>2.1.1. Supervised Learning</b> .....	35
<b>2.1.2. Unsupervised Learning</b> .....	44
<b>2.3. Deep Learning</b> .....	50
<b>2.4. Feature Engineering</b> .....	62
<b>2.4.1. Feature Aggregation for Evolving Customer Behaviour</b> .....	62
<b>2.4.2. Feature Transformation using Mathematical Equations</b> .....	66
<b>2.5. Feature Selection</b> .....	73
<b>2.6. Feature Engineering Tools and Framework</b> .....	77
<b>2.7. Summary &amp; Conclusion</b> .....	83
<b>2.8. Literature Synthesis</b> .....	85
<b>3. Feature Engineering Framework for Financial Fraud Detection Models</b> .....	87
<b>3.1. Overview</b> .....	87
<b>3.2. Feature Creation Processes</b> .....	89
<b>3.2.1. Data Preparation</b> .....	89

<b>3.2.2. Feature Creation Processes</b> .....	103
<b>3.3. Feature Selection Process Component</b> .....	108
<b>3.3.1. Feature Selection</b> .....	109
<b>3.3.2. Performance Metrics for Fraud Detection Models</b> .....	116
<b>3.4. Key Summary</b> .....	133
<b>4. Online Banking Transaction Data</b> .....	134
<b>4.1. Data Source and Description</b> .....	134
<b>4.2. Exploratory Data Analysis (EDA)</b> .....	138
<b>4.3. Conclusion</b> .....	148
<b>5. Experiments and Validation of Fraud Detection Framework</b> .....	149
<b>5.1. Data Preparation processes</b> .....	149
<b>5.2. Feature Creation Processes for Experiment</b> .....	154
<b>5.3. Feature Selection Processes for the Experiment</b> .....	157
<b>5.4. Model Preparation</b> .....	160
<b>5.4.1. Split the dataset into Training and Test sets</b> .....	160
<b>5.4.2. Modelling</b> .....	162
<b>5.5. Model Preparation</b> .....	168
<b>5.6. Evaluation and Discussions</b> .....	188
<b>5.7. Conclusion</b> .....	194
<b>6. Conclusion and Future Work</b> .....	195
<b>6.1. Introduction of the main achieved work</b> .....	195
<b>6.2. Research Contributions</b> .....	201
<b>6.3. Recommendation for Future Research</b> .....	202
<b>References</b> .....	204

## **Lists of Figures:**

Figure 1-1. Annual remote banking fraud losses

Figure 1-2. Annual case volumes of remote banking fraud

Figure 1-3. Internet banking fraud losses

Figure 1-4. Example of a feature engineering concept

Figure 1-5. Example of the effect of feature engineering

Figure 1-6. A flow for building an effective classification ML/DL model

Figure 1-7. Overview of the approach steps in the thesis

Figure 2-1. Support vector machine approach

Figure 2-2. The maximum margin hyperplane

Figure 2-3. Hybridization of supervised and unsupervised learning

Figure 2-4. Decision tree for fraud detection

Figure 2-5. Random forest

Figure 2-6. Artificial neural network architecture

Figure 2-7. Activation function for NNs

Figure 2-8. Layer of ANNs in credit card

Figure 2-9. Image of grouping customers in similar data pattern

Figure 2-10. K-means clustering

Figure 2-11. Anomaly detection with isolation forest

Figure 2-12. Example of isolating a non-anomalous point in a 2D gaussian distribution

Figure 2-13. Example of isolating an anomalous point in a 2D gaussian distribution



Figure 2-14. The logic of local outlier factor

Figure 2-15. Deep learning neural networks

Figure 2-16. Autoencoder with hidden layers

Figure 2-17. Autoencoder neural networks

Figure 2-18. Convolutional neural networks architecture

Figure 2-19. Basic structure of CNNs

Figure 2-20. Recurrent neural networks with a hidden state

Figure 2-21. RNNs architecture

Figure 2-22. The concepts of RNNs with LSTM

Figure 2-23. Feature aggregation for evolving customer behaviour

Figure 2-24. Applying the transaction aggregation process with the HOBA principle

Figure 2-25. Example of how to deal with the image recognition data in deep learning processes

Figure 2-26. Connections between the nodes with each weight

Figure 2-27. The impact of feature scaling on the wine dataset

Figure 2-28. Feature Transformation using PCA

Figure 2-29. Overfitting the training data

Figure 2-30. Underfitting the training data

Figure 2-31. Appropriate fitting model

Figure 2-32. Demonstration of the concept deep feature synthesis

Figure 2-33. The algorithm of deep feature synthesis

Figure 2-34. Sample case of using DFS tool

Figure 2-35. Features creation using the DFS tool

Figure 2-36. ExploreKit system architecture

Figure 3-1. Conceptual feature engineering framework

Figure 3-2. Example of different sources

Figure 3-3. Image of data integration into the banking system

Figure 3-4. Logical data modelling of online banking system

Figure 3-5. Example of one-hot encoding method

Figure 3-6. Sample of dummy coding

Figure 3-7. Specific number can connect to other information tables

Figure 3-8. Blanks indicate missing values in dataset

Figure 3-9. A flow of processes for dealing with missing values

Figure 3-10. Standardisation feature values

Figure 3-11. Sample of correlation matrix

Figure 3-12. Outline processes in the random forest algorithm

Figure 3-13. Decision trees inside of random forests

Figure 3-14. Feature importance scores of top 30 features

Figure 3-15. Linear and nonlinear separation of sample data

Figure 3-16. Mapping the data from two-dimensional space to three-dimensional space

Figure 3-17. Nonlinear SVM with polynomial kernel

Figure 3-18. SVM classifier using an RBF kernel

Figure 3-19 One-class support vector machine

Figure 3-20. The steps of random forest algorithm

Figure 3-21. Instance of LOF

Figure 3-22. Architecture of autoencoder

Figure 3-23. Precision versus recall

Figure 3-24. Example of a precision and recall curve

Figure 3-25. Typical ROC curve

Figure 3-26. Area under the ROC curve (AUC)

Figure 4-1. ER diagram for an online banking system

Figure 4-2. Unbalanced dataset

Figure 4-3. The distribution of transaction amount

Figure 4-4. Boxplot of transaction amount by fraud frag

Figure 4-5. Distribution of the transaction frequency of the available balance

Figure 4-6. Distribution of transaction login latency

Figure 4-7. Access device type used for transaction

Figure 4-8. Credit card types

Figure 4-9. Comparison between normal and fraud patterns in access code types

Figure 4-10. Various types of client's screen browser in IDVD\_CLIENTSCREENRESOID

Figure 4-11. Transactions over timestamp in months

Figure 4-12. Transaction timestamps in the dataset

Figure 4-13. Transactions over timestamp in weekdays

Figure 4-14. Transactions over timestamp in days

Figure 4-15. Transactions over timestamp in hours

Figure 4-16. Distribution of log transformation amount

Figure 4-17. Distribution of log transformation available balance

Figure 4-18. Login latency transformed by standard deviation

Figure 5-1. Training set and test set

Figure 5-2. A way how to consider splitting a dataset

Figure 5-3. The dataset split with 80:20 ratio

Figure 5-4. AUC of the one-class SVM model with dataset 1

Figure 5-5. AUC of the one-class SVM model with dataset 2

Figure 5-6. AUC of the one-class SVM model with dataset 3

Figure 5-7. AUC of the RF model with dataset 1

Figure 5-8. AUC of the RF model with dataset 2

Figure 5-9. AUC of the RF model with dataset 3

Figure 5-10. AUC of the Isolation Forest model using dataset 1

Figure 5-11. AUC of the Isolation Forest model using dataset 2

Figure 5-12. AUC of the Isolation Forest model using dataset 3

Figure 5-13. AUC of the LOF model using dataset 1

Figure 5-14. AUC of the LOF model using dataset 2

Figure 5-15. AUC of the LOF model using dataset 3

Figure 5-16. The reconstruction error threshold 4 in dataset 1

Figure 5-17. Confusion matrix of the autoencoder model with dataset 1 using a threshold of 4

Figure 5-18. AUC of the AE model of threshold 4 with dataset 1

Figure 5-19. The reconstruction error threshold 4 in dataset 2

Figure 5-20. Confusion matrix of the autoencoder model with dataset 2 using a threshold of 4

Figure 5-21. AUC of the AE model of threshold 4 built with dataset 2

Figure 5-22. The reconstruction error threshold 4 in dataset 3

Figure 5-23. Confusion matrix of the autoencoder model with dataset 3 using a threshold of 4

Figure 5-24. AUC of the AE model of threshold 4 built with dataset 3

Figure 5-25. The reconstruction error threshold 1 in dataset 1

Figure 5-26. Confusion matrix of the autoencoder model with dataset 1 using a threshold of 1

Figure 5-27. AUC of the AE model of threshold 1 built with dataset 1

Figure 5-28. The reconstruction error threshold 1 in dataset 2

Figure 5-29. Confusion matrix of the autoencoder model with dataset 2 using a threshold of 1

Figure 5-30. AUC of the AE model of threshold 1 built with dataset 2

Figure 5-31. The reconstruction error threshold 1 in dataset 3

Figure 5-32. Confusion matrix of the AE model with dataset 3 using Threshold of 1

Figure 5-33. AUC of the AE model of threshold 1 built with dataset 3

Figure 6-1. The outputs in each phase

## Glossary:

UK	United Kingdom
US	United States
ATM	Automated Teller Machine
ML	Machine Learning
FDS	Fraud Detection Systems
SVM	Support Vector Machine
NN	Neural Networks
DT	Decision Tree
RF	Random Forests
HMM	Hidden Markov Model
PGA	Peer Group Analysis
RNN	Recurrent Neural Network
DL	Deep Learning
FE	Feature Engineering
ANN	Artificial Neural Networks
PCA	Principle Component Analysis
IF	Isolation Forests
LOF	Local Outlier Factor
AE	Autoencoder
CNN	Convolutional Neural Networks
AUC	Area Under the ROC Curve
ID	Identification
IP	Internal Protocol Address
OHE	One-Hot Encoding
ReLu	Rectified Linear Unit
ELU	Exponential Linear Unit
SGD	Stochastic Gradient Descent
TP	True Positive

TN	True Negative
FN	False Negative
FP	False Positive
TPR	True Positive Rate
FPR	False Positive Rate
EDA	Exploratory Data Analysis
ROC	Receiver Operating Curve

# 1. Introduction

As online payment system advances, fraud schemes have shifted from physical fraud actions by using stolen credit cards directly or into online banking fraud actions by using advanced digital techniques. Until several years ago, the use of credit or debit cards were the great majority of transaction methods in financial services via ATMs, shops, and at bank teller. However, in recent years, remote online banking using the internet via any portable devices has become a popular method for transaction money and at the same time, financial fraud losses via the online banking have been increasing. According to UK finance report in 2021 [1] (see Figure 1-1), total losses through remote online banking in the United Kingdom have reached £197.3 million in 2020, up 31percent higher than in 2019. The annual number of cases of fraudulent transaction via internet banking and mobile banking has been rapidly growing from 32,721 cases in 2019 to 66,150 cases in 2020, with other financial fraud losses such as payment cards and cheques decreasing from £470.2 million to £452.6 million [1] (see Figure 1-2).

Remote banking values	2013	2014	2015	2016	2017	2018	2019	2020	19/20 % Change
Internet banking	£58.8m	£81.4m	£133.5m	£101.8m	£121.2m	£123.0m	£111.8m	£159.7m	43%
Telephone banking	£13.1m	£16.8m	£32.3m	£29.6m	£28.4m	£22.0m	£23.6m	£16.1m	-32%
Mobile banking	N/A	N/A	£2.8m	£5.7m	£6.5m	£7.9m	£15.2m	£21.6m	41%
<b>TOTAL</b>	<b>£71.7m</b>	<b>£71.9m</b>	<b>£98.2m</b>	<b>£137.0m</b>	<b>£156.1m</b>	<b>£152.9m</b>	<b>£150.7m</b>	<b>£197.3m</b>	<b>31%</b>

Figure 1-1. Annual remote banking fraud losses [1]



Annual case volumes Remote Banking fraud 2013-2020									
Remote banking cases	2013	2014	2015	2016	2017	2018	2019	2020	19/20 % Change
Internet banking	13,799	16,041	19,691	20,088	21,745	20,904	25,849	55,995	117%
Telephone banking	5,596	5,578	11,380	10,495	9,577	7,937	11,199	7,490	-33%
Mobile banking	N/A	N/A	2,235	2,809	3,424	2,956	6,872	10,155	48%
<b>Total</b>	<b>19,395</b>	<b>21,819</b>	<b>33,306</b>	<b>33,392</b>	<b>34,746</b>	<b>31,797</b>	<b>43,920</b>	<b>73,640</b>	<b>68%</b>

Figure 1-2. Annual case volumes of remote banking fraud [1]

Specifically, internet banking fraud cases have been increasing the most. Over 72 percent of UK people utilised online banking in 2019 and the number of online banking user have further increased in 2020. Thus, the number of fraudulent schemes using internet banking have increased and the number of losses by the internet banking fraud tends to increase year by year (see Figure 1-3). The internet banking fraud has been taking place when a fraudster gets access to a customer’s account of online banking and carries out an unauthorised money transfer [1].

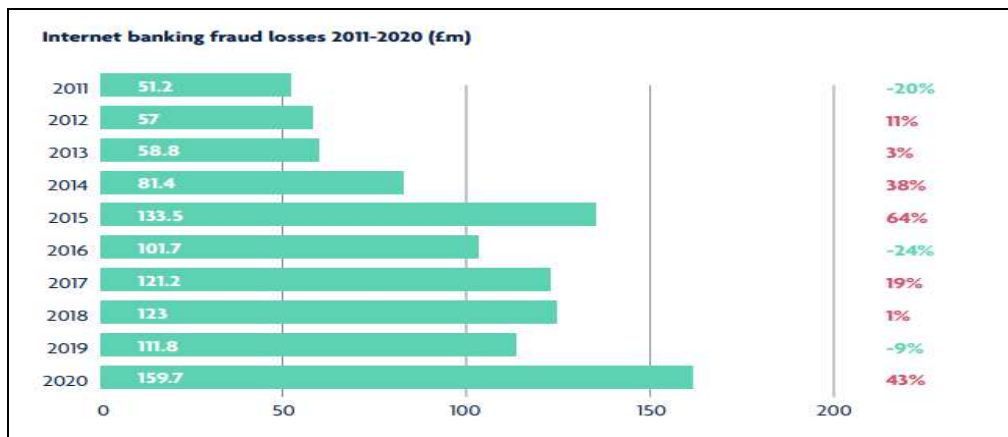


Figure 1-3. Internet banking fraud losses [1]

Fraudster have stolen money by accessing customer’s bank account via the remote banking channels. The fraud schemes have become digital and technologically more advanced. To address constant changes in fraud behaviour, in recent years, many

academic studies and some financial industries proposed to employ machine learning (ML) methods in fraud detection systems (FDS) [2][3]. A variety of ML algorithms are used for detecting fraud patterns from the large amounts of transaction dataset. For instance, support vector machines (SVM) [4], neural networks (NN) [5][6], decision tree (DT) [7] and random forests (RF) [8] are popularly used in many studies and they improved the accuracy of fraud detection models. Other often-used techniques for discovering a fraud based on user's regular behaviour in real time are the Hidden Markov Model (HMM) [9], Peer Group Analysis (PGA) [10], a recurrent neural network (RNN) [11] [12], and deep learning [13] [14] [15]. However, it is still challenging to detect new fraudulent actions by only using raw feature values in original dataset.

A feature represents a measurable aspect of raw data that can be utilised for analysis [17]. Features come from available data and appear as columns in datasets such as Name, Postcode, Phone number, Age, Sex, and so on. Performance of machine learning models depends on the quality of the features [22]. Feature engineering (FE) is known as techniques to create new features by transforming raw features or to extract effective features from different sources [17] [104]. In the past several years, most studies of using feature engineering have been image and voice recognitions which deal with high-dimensional features such as high resolution of images (pixels) and digital audio files (mp3, etc) [16] [17] [18]. In recent years, feature engineering became a popular method in classification studies and has been implemented for improving an accuracy of machine learning models [19] [20] [21]. The aim of using FE for classification is to reveal latent data patterns from an original dataset, which enable ML algorithms to learn differences in a target. For instance, Figure 1-4 illustrates two different depictions for points belonging to a classification problem dataset. On the left, one can see that instances in

connection with the two classes are present in alternating small clusters. It is difficult to draw a reasonable classifier on this depiction that separates the two classes for the most machine learning algorithms. On the other hand, if the feature  $x$  is replaced by its *sine*, as seen in the depiction on the right, it makes the two classes reasonably separable by most classifiers. The process of altering the feature representation of a prediction modelling problem to better fit a training algorithm is called “Feature Engineering” [22].

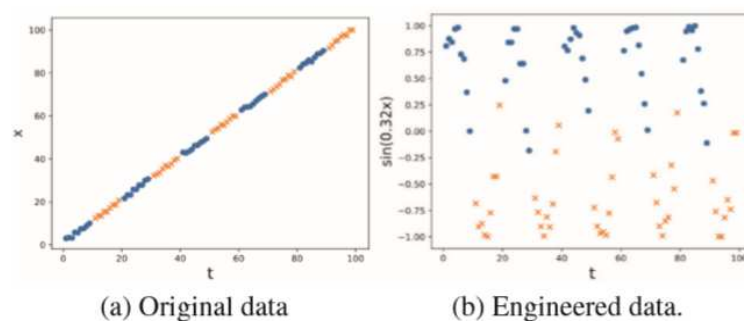


Figure 1-4. Example of a feature engineering concept [22]

Feature engineering is a method of mapping a given data into a different dimension which is easier to interpret for a ML algorithm and is considered as a key method to improve accuracy of the models of ML and DL by transforming the given data into new features which represent latent data patterns. The most well-studied feature engineering method in financial fraud detection for creating new features is feature aggregation. Feature aggregation is to create new features which are a combination of two or more features in the given data using domain knowledge to create a new feature. For instance, in case of transaction fraud, if a fraudster uses the stolen credit card in unusual hours and transaction amount unlike the credit card holder, it may be possible to differentiate between a fraud and the card holder by a fraud detection model. However, if there is not really a difference in individual features such as time and amount, it will not be easy to distinguish a fraud by using only the raw features for a fraud detection model. New features, which aggregate

two or more individual features in a dataset, will give a clear boundary to algorithms for easily understanding and reveal the latent pattern of user behaviour on transaction data.

Feature transformation is another technique for mapping raw data to a different space and uses mathematical or statistical functions on raw data to create new features. The new created features will not hold the same interpretation as the original features, but they will obtain more distinguishable capability in a different dimension than the original dimension. For Instance, Figure 1-5 shows before and after implementing a feature engineering method on the raw data. On the left hand side plot, there are two classes of points in a circular pattern. For any algorithms, it is not easy to divide this data pattern into one part. On the right hand side plot, the input data is clearly divided into two parts for the algorithms.

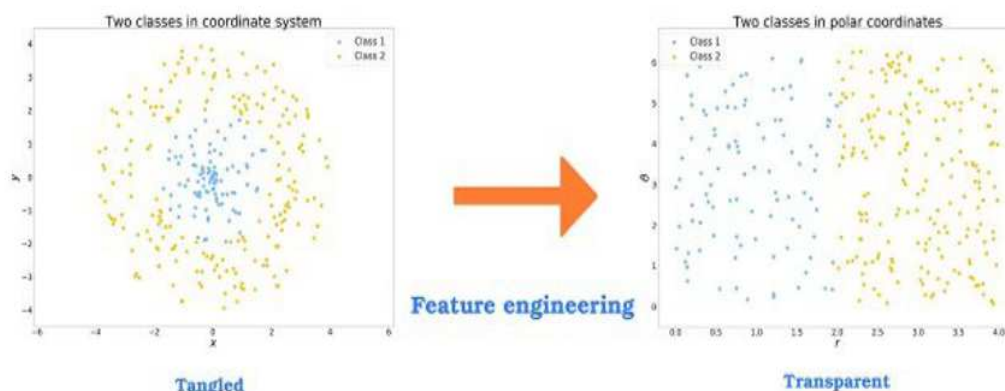


Figure 1-5. Example of the effect of feature engineering [22]

In order to transform the given data to make the algorithms understand easily the different patterns, a cartesian coordinate system ( $x^2 + y^2 = r^2$ ) was applied to the data. The data was transformed to be understandable for the algorithms.

Not a lot of research has been carried out using FE for improving an accuracy of a fraud detection model. A few studies using feature engineering for classification problem

implemented feature engineering methods on the original dataset and created new engineered features which were used for building a machine learning model. They evaluated the model performance with/without the engineering features and demonstrated the effectiveness of using feature engineering [19] [20]. Although these studies introduced the methods on how to create new features using feature engineering and used all created features, they did not select features after the creation. In general, the performance of machine learning algorithms is influenced by input feature values for better or worse. Thus, containing lots of features will cause worse performance of machine learning models, such as overfitting and low accuracy.

Overfitting occurs when a machine learning model learns all the data points in the given dataset [80]. Hence, involving many irrelevant features in training data will give a negative impact on classification accuracy of a fraud detection model because of a noise in the learning process. Therefore, there exists some research in financial fraud detection using feature engineering and they introduced a feature engineering as feature selection techniques for selecting better attributes from the dataset only.

As a result, in this research I proposed and studied a feature engineering framework which consists of two methods namely feature creation and feature selection. This research is mainly focused on the possibility to develop an innovative feature engineering framework to create and select the effective features for online banking fraud detection. The contribution of my research will be three-folds. First, to develop a new feature engineering framework that can provide an effective feature set for improving an accuracy of a financial fraud detection model. Second, to use both feature engineering and feature selection methods simultaneously in the framework. Third, to use the actual

online banking dataset including fraudulent data and examine how the provided feature set from the framework helps to improve the performance of fraud detection models.

## **1.1. Research Problem**

As part of my hypothesis the key research question I came up with is: if an effective features set is created by using all methods of feature aggregation and feature transformation and feature selection consistently in a framework, the performance of fraud detection models built with effective features set is likely to be improved.

In general, the performance of machine learning models is determined by the quality of training data. Specifically, raw data collected from various sources is messy and disorganised in the first stage. In the raw data, some feature values have missing values and character strings that machine learning algorithms cannot handle. Therefore, the raw data needs to be cleaned and converted to numerical data prior to implementing feature engineering. According to the relevant studies for classification cases using feature engineering, they have used the definition of feature engineering differently. Many studies have defined feature engineering as feature creation and created new features by aggregating individual features in dataset. In other studies, feature engineering is defined as feature selection or extraction methods for selecting appropriate and effective features from different data sources.

This study will address the research questions presented below:

- Can new features from original dataset be created by defining feature aggregation and transformation methods? (See Section 3.2.2)

- Can effective features be selected to improve performances of machine learning and deep learning models from the whole data? (See Section 3.2.3)
- Can a model based on the created features set by the feature engineering framework improve the performance of machine learning models? (See Section 5.4)
- Can a model's performance be improved by using the features set which is created by using all methods which include feature aggregation, feature transformation and feature selection consistently in the framework rather than the model build with original features set? (See Section 5.5)

The outcomes of this research will be used to provide better protection to the banking customers by using more effective feature set which improves the performance of ML and DL models in fraud detection systems.

## **1.2. Aim and Objectives of this research**

The aim of this research is to develop feature engineering framework which can produce a new feature set used for machine learning and deep learning models to learn transaction behaviour, that will improve accuracy of fraud detection and decrease the number of losses by fraudulent transaction via online banking. This will be achieved by creating a novel framework which generates new features by effective feature engineering methods and selects appropriate features that can improve the accuracy of ML and DL models. One will focus on the development of feature engineering framework that can create useful features that will influence the accuracy of fraud detection models by using feature engineering techniques in combination with feature aggregation and transformation. Combining both techniques will lead to a high potential features that can improve ML/DL

model's performance. The effectiveness of the fraud detection models is determined by either the volume of data or other qualities of the data. The two main elements on the framework – feature creation and feature selection are important processes to produce effective features set. Including many redundant or high correlated features in the process of training data will have a negative influence on classification of ML models in terms of overfitting. To reduce the overfitting risk, I added the feature selection processes in the framework as a part of feature engineering. The feature selection component contains two major feature measurement methods: correlation covariance and feature importance.

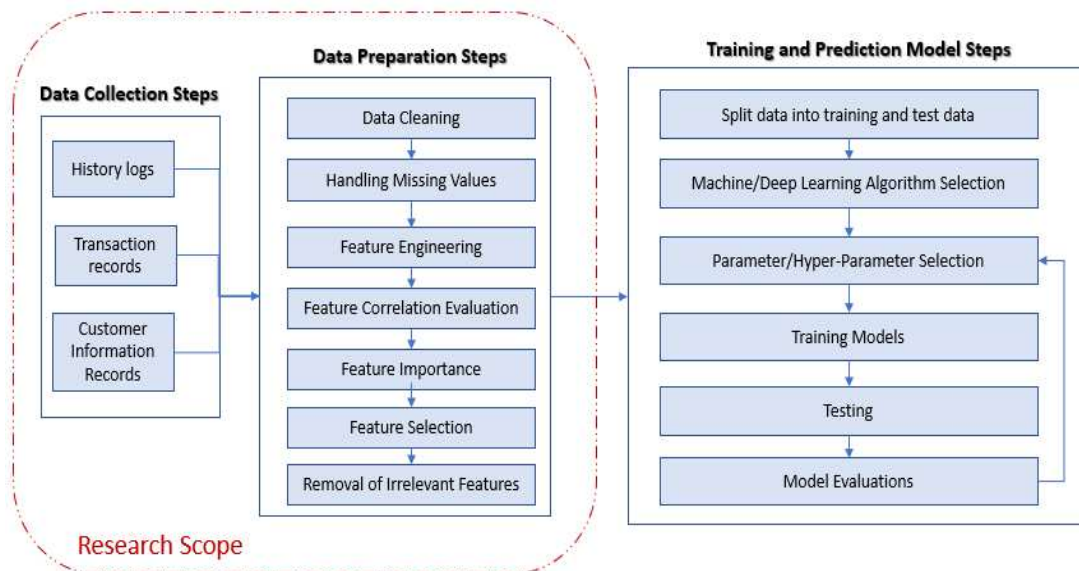


Figure 1-6. A flow for building an effective classification ML/DL model  
(This is further explained in Chapter 3)

Figure 1-6 shows a flow for building an effective classification machine learning or deep learning model. This describes a whole structure of steps in data preparation and training ML/DL models. Both steps of data preparation and training and prediction model have the potential for improving the performance of a fraud detection model. In the case of financial fraud detection, many studies have focused on the training and prediction model steps by adjusting parameters in the selected ML/DL algorithms or building a new



prediction model with a novel concepts. On the other hand, the number of studies that have mainly dealt with the data preparation portion including feature engineering for financial fraud detection is very few . Therefore, my research scope focuses on the data preparation phases which consists of data cleaning, feature engineering and feature selection. The prepared dataset that is used for training a ML or DL model have still a chance to make the models improve to a better accuracy.

The main objectives of this research are:

- To explore the current state of research in fraud detection and the cases specifically using feature engineering methods for classification models, and to identify the main issues, existing approaches, and available methods for improving performance of the fraud detection models. See Chapter 2 for detailed exploration.
- To investigate database structure tables of banking transaction and to consider which attributes in each table are constantly available to be extracted. See Chapter 3 for concrete idea of which attributes commonly exist on banking data.
- To investigate how to deal with character string datatype values and missing values in each attribute. See Chapter 3 for exploration of appropriate techniques.
- To research into both methods of feature engineering and selection for fraud detection and to consider how to create new features that express customer's behaviour during a transaction and reveal the different aspect of input values for making machine learning or deep learning models distinguish between normal and fraud easier. Also, to consider how to select the effective features from all attributes. See Chapter 3 for a detailed methods of creating and selecting effective features.

- To design a new framework that provides an effective feature set which can be prepared by both in terms of feature engineering and feature selection. See Chapter 3 for more details.
- To analyse the multidimensional banking dataset which was provided by a private European Bank in terms of both the exploratory data analysis with visualisation and the assessment of available attributes in the dataset. See Chapter 4 for analysis of the given data.
- To implement the feature engineering framework by using the actual banking dataset for evaluating the effectiveness of a use of the engineered features. Therefore, the models are built by the different types of the datasets that are the raw dataset, all features set, and the selected features set. The model's performances are evaluated by some classifier metrics. See Chapter 5 for performance evaluation by using appropriate metrics.

### **1.3. Research Methodology**

The methodology of the proposed research combines some methods, which are needed to fulfil individual objectives. The methodology consists of three different phases: analytical approach, practical approach, and observational approach, to build a new feature engineering framework that can provide an effective features set for various machine learning algorithms. In the analytical approach, there are mainly three key topics which are needed to analyse: retrievable data, feature engineering methods, and ML and DL algorithms commonly used as a fraud detection model. Scenarios for feature generation without considering what data is consistently obtainable may not be generalizable. First, that is because it may not be possible to obtain the expected feature values for applying

feature aggregation. Second, it is necessary to understand the current situation of feature engineering methods and frameworks for finding out the gap between the current methods and hypothesized methods in my research. Lastly, investigating what algorithms are commonly used as a fraud detection model is important because a new dataset prepared through the processes in the framework needs to improve the performance of any algorithms of the fraud detection model. The analytical approach will mainly be applied during the research stage when there is a need for crucial analysis and comparison of other methods. In practical approach, there are three topics: exploratory data analysis (EDA), framework development and adjustment. EDA is to visualise trends and patterns in the obtained data. Before applying EDA using the data directly in the feature engineering processes, it is important to grasp the data tendency of what data is dealt with in the experiment. Based on the analysis of the feature engineering methods in the analytical approach, the feature engineering framework is developed and implemented. During this phase, it is significant to apply the planned feature engineering methods to the actual data obtained, understand the gaps between the resulting values, and then adjust the method if necessary. The observation approach will be applied for measuring the effect of the created feature set by comparing the performance of each machine and deep learning models and evaluating the use of the feature engineering framework. The objective of the observational approach is to determine whether the datasets generated are valid based on metrics that can accurately assess the performance of fraud detection models based on imbalanced data.

### **1.3.1. Analytical Approach**

Before collecting data, it is important to consider a scenario of customer's behaviour during a transaction and which data is available to extract. Many studies on financial fraud

detection focused on how to create effective features for a machine learning algorithm without considering whether data is available or not. However, in this research, analysing what data is available on online banking system is essential because this makes it possible to generate features that have a certain effect regardless of which financial institution uses them.

Therefore, it is necessary to grasp what data can be obtained from a typical bank system, and to conceptualize what kind of feature values related to customer behaviour can be generated from those data. That is the analytical approach.

■ **Data Collection** – collect data from various sources which are connected to transactional banking database. The collected data will need to be arranged for creating valuable features that will improve accuracy of fraud detection models. Two different categories of data will be collected:

- Dynamic data- shows user activities or behaviour on online banking and via ATM machine, such as transaction records, amount, location, IP address, latency, access record, timestamp, etc and will be frequently updated.
- Static data - indicates general information with relation to customer and bank information, i.e., name, age, birthday, email, home address, phone number, bank account, etc and will be seldom updated.

■ **Data Preparation** – considering how to deal with missing data and the type of character strings will be demanded because the collected data from various sources usually has a lot of missing variables and the different types of data. Inappropriate data processing will cause lowering the prediction accuracy of machine learning models. This will focus on using technique of dealing with missing values and

converting the type of character strings to numeric types correctly and efficiently. Therefore, this will provide deeper insight into data preparation for fraud detection.

- **Feature Engineering Exploration** – investigate techniques of feature engineering for classification between feature aggregation and transformation. Feature aggregation indicates how to create new features which can show the difference between normal and fraudulent behaviour on transaction. Feature transformation indicates how to create new features which will represent latent pattern of the raw data by applying various mathematical functions. Also, this will be needed to study the feasibility of these techniques on general transactional banking dataset.
- **Feature Evaluation Exploration**– investigate the correlation between feature creation and selection in terms of the effectiveness of impact on prediction accuracy of ML models. And then, determine methods on how to evaluate and select the features in the dataset.
- Analytical approach is further expanded in Chapter 3 and 4.

### **1.3.2. Practical Approach**

- **Framework Development and Implementation**– This is to investigate available toolsets and a software for developing the framework that can provide a fundamental platform to achieve my objectives. This will include the data preparation such as data cleaning and handling missing values. Also, investigation on which machine learning and deep learning algorithms best fit the needs for evaluating the effectiveness of the created feature set will be addressed.

- **Model Development** – This is to build fraud detection models by training each algorithm with the three different types of datasets: an original dataset, a dataset including all created new features by feature engineering and the selected features dataset.
- Practical approach is further expanded in Chapter 5.

### **1.3.3. Observation Approach**

- **Performance Evaluation and Analysing** – both performance of ML/DL models and the effectiveness of using the feature set prepared by the framework will be evaluated with respect to the predictions with the actual data. Various comparison of the performance between the different types of models will be analysed.
- Observation approach is covered in detail in Chapter 5.

## 1.4. Summary Overview Diagram

The below diagram illustrates an overview of the approaches for the research. In the thesis, I proceed each phase in order.

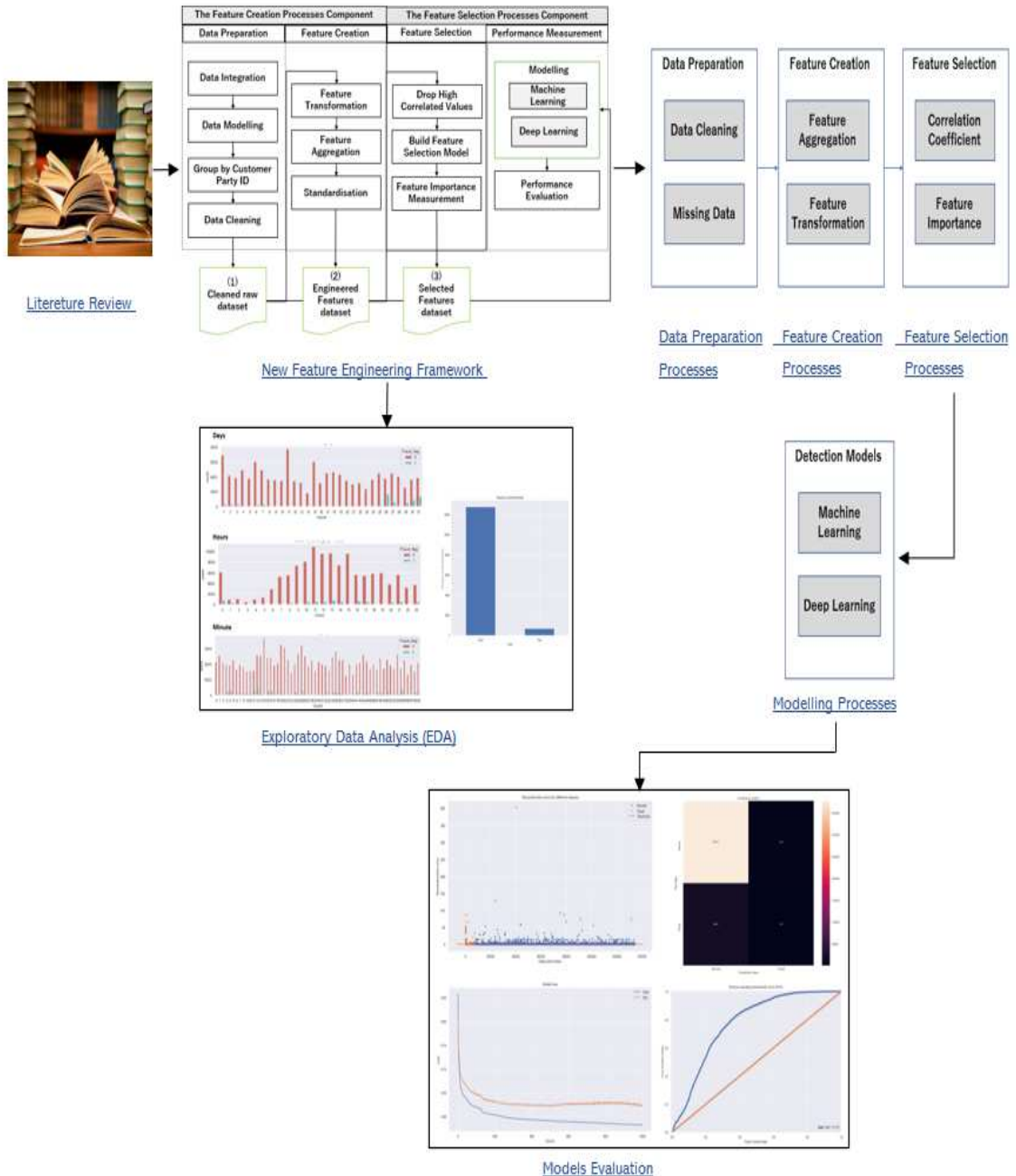


Figure 1-7. Overview of the approach steps in the thesis

## 1.5. The Contributions to Knowledge

The research conceptualizes a feature engineering framework for financial fraud detection that creates an effective feature set for either machine learning or deep learning models based on customer's behaviour and information on transactions. Additionally, feature transformation methods which are popularly utilised for dealing with high-resolution data in image processing are incorporated to the feature creation process in the framework for financial fraud detection. Furthermore, it also covers feature selection process that remove unnecessary features from the dataset to avoid overfitting or a negative influence on the performance of models in the framework. In many studies for fraud detection using feature engineering, the three methods of feature aggregation and feature transformation and feature selection were not incorporated consistently. Particularly, there were not feature engineering methods for preparing better input data for deep learning in classification. In my research hypothesis, a features set created by the processes in the framework will enable ML/DL models to detect fraudulent transaction with better accuracy than the models without the features. More specifically, the research contributions include:

- Develop a new feature engineering framework that consists of major three feature engineering methods namely feature aggregation, feature transformation and feature selection consistently. There are not many features engineering-based frameworks in financial fraud detection. Few frameworks exist for financial fraud detection; however, they are only based on feature creation but not feature selection.
- Build a new feature engineering creation method which combines both feature aggregation based on customer behaviour and feature transformation for revealing



new data patterns. Before this research, there were no feature creation method using both different feature engineering of feature aggregation and feature transformation. In particular, feature transformation was not used in the financial fraud detection case and was mainly used in image processing.

- Demonstrate the effectiveness of features prepared through the processes and techniques in the framework by using an actual online banking transaction dataset and comparing the performance between a baseline model and a model built with an optimised feature set.
- Generate the feature engineering framework for both machine learning and deep learning that are popularly used in fraud detection cases. It was proved that both algorithms have a common effect using the created features set based on real-life bank transactions. Before this research and in many relevant studies either machine learning or deep learning were used but not both.

## **1.6. Structure of the thesis**

To report the findings of the research in detail, this document is organized as follows:

### **Chapter 2: Literature Review**

In this chapter, various research papers and articles have been reviewed and studied in the context of the feature engineering, some algorithms of machine learning and deep learning for fraud detection. The description of the existing research, methodologies and proposed solutions to problems closely related to feature engineering, and each algorithm how to deal with input values were performed.

### **Chapter 3: A Feature Engineering Framework for Financial Fraud Detection Models**

This chapter forms the core of the thesis. It provides details about the conceptual framework of feature engineering for fraud detection. The core focused has been on methods of feature engineering where feature creation and selection are discussed. This is followed by fraud detection models that provide insight into handling the input data and learning capabilities and are discussed with the appropriate metrics for performance evaluation.

### **Chapter 4: Online Banking Transaction Data**

This chapter provides exploratory data analysis using an actual banking transaction dataset and summarize the key observations based on the exploratory data analysis.

### **Chapter 5: Experiments and Validation of the Framework**

Based on the conceptual framework discussed in Chapter 3, this chapter provides experimental approaches using the actual online transaction dataset that is explained in Chapter 4. The chapter also evaluates and compares the performances of the models based on appropriate criteria.

### **Chapter 6: Conclusion and Future Work**

The final chapter provides conclusion on the thesis, describes what has been achieved, recaps on the research contribution and recommendations for the further work.

## **2. Literature Review**

Research hypothesis states that the performance of either machine learning or deep learning for financial fraud detection can be improved if effective features set that is created by using all methods of feature aggregation and feature transformation and feature selection is consistently used in the framework. The relevant literature review was conducted to support the research hypothesis and to produce a novel feature engineering for financial fraud detection.

### **2.1. Machine Learning Algorithms**

Many different methods/algorithms of machine learning (ML) algorithms are used for detecting fraudulent transactions across a variety of areas such as a credit card, online payment and remote banking. Both supervised learning and unsupervised learning are popularly used as a fraud detection model in a variety of the related research papers.

#### **2.1.1. Supervised Learning**

Supervised learning uses training data including desired outputs which is also called a target for learning the data patterns and it is typically used for classification such as fraud detection (fraudster or customer).

Here are some of the popular supervised learning algorithms used in the studies of fraud detection.

##### **I. Support Vector Machines (SVMs)**

SVMs are a popular supervised learning models that classifies input samples as fraud or not in some studies of fraud detection [4][27][28]. SVM for binary classification uses a

boundary line, which is called an optimal hyperplane, to separate the two classes between fraud and non-fraud. The boundary line does not only separate the classes but also settles as far away from the closest samples between two features:  $x_1$  and  $x_2$  as possible as shown in Figure 2-1. The distance between the optimal hyperplane and the closest data point from the two classes is called the margin. SVM calculates the maximum margin by using the Euclidian norm to determine an optimal hyperplane in dimensions greater than 2 for better classification. For instance, in the two features of  $x_1$  and  $x_2$ , the best hyperplane can separate two classes (labelled 'o' and '◇' shown in Figure 2-1) between the points of both closest samples.

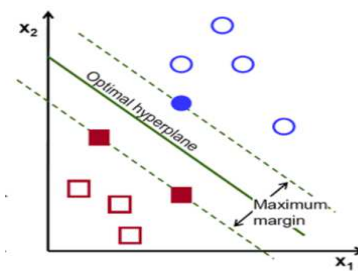


Figure 2-1. Support vector machine approach [89]

A linear classifier is defined as  $h(x) = \sin n(wTx + b)$  and it is assumed a binary classification setting with labels  $\{+1, -1\}$ . The margin ( $\gamma$ ) is the distance from the hyperplane to the closest samples.

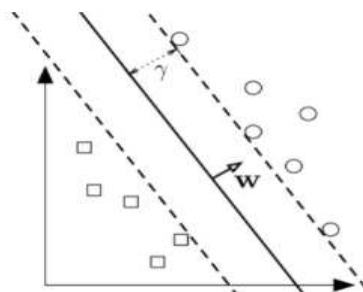


Figure 2-2. The maximum margin hyperplane [90]

A hyperplane is determined by a vector  $w$  which is the normal vector. The hyperplane is described as  $H = \{x | w^T x + b = 0\}$ . The equations of  $w^T x - b = 1$  (anything is one class) and  $w^T x - b = -1$  (anything is other class) are calculated to determine the best position in Figure 2-2.

D. Abdelhamid et al. [4] suggested the automated fraud detection system using SVMs for detecting various type of banking fraudulence. In the study, they reinforced both supervised learning and unsupervised learning by hybridizing two classes as illustrated in Figure 2-3. Their technique is to separate fraud transaction from non-fraud transactions to avoid a false generalisation by filtering the positive part. Learning only single class could well reduce non-fraud transactions space and extend the space of fraud transactions by using hybridisation. They built the SVM models with three different datasets for testing various types of frauds: credit card fraud, money laundering and mortgage. Based on the hybridisation of single class and binary SVM methods, the performance of the models was significantly improved compared to similar studies using other algorithms.

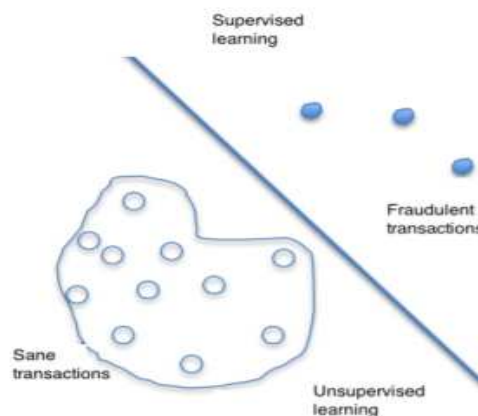


Figure 2-3. Hybridization of supervised and unsupervised learning [27]

D.Zang et al. [27] also used a weighted SVM technique to detect credit card fraud. The SVM technique considers the cost of misclassification by using transaction balances.

Weights in the paper were introduced as the penalty of misclassification and reflected the financial importance between two classes of fraud and non-fraud transactions. Credit card fraud dataset is usually imbalanced. The weighted SVM model with random under sampling method as a weight for data points of fraud was built and applied to samples of credit card transactions in a European bank. The result described that using the weighed SVM technique could dramatically improve the detection models built with other algorithms. However, the most difficulties of using SVM are selecting appropriate hyper parameters and dealing with larger data sets.

## **II. Decision Trees (DT)**

Decision Trees (DT) is a supervised learning method for classification, and it is used to visually represent decisions by making processes such as tree structure. DT generally begins with a single node, where diverges in possible outcomes. In the trees, each node represents target labels while the outcomes represent junctions of features bringing to target labels. Figure 2-4 shows the diagram of a possible decision tree for detecting fraudulent action based on several input features. Each question of features is the node. The judgements of “yes” or “no” draw the branches in the tree to the next child nodes. If the larger amount than average was spent in 24 hours and several purchases were occurred in a day from risky venders, it is classified as fraud with 90 percent probability.

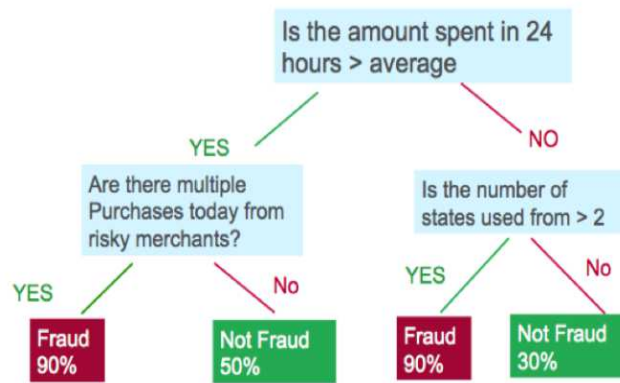


Figure 2-4. Decision tree for fraud detection [7]

DT decides the optimal choice based on purity by calculating the information gain for splitting nodes with the highest value. The information gain is measured by entropy which is the criterion of impurity which leads to how a DT model decides to split the data. The entropy equation is as below:

$$E(S) = \sum_{i=1}^c - p_i \log_2 p_i \quad (\text{eq. 2.1})$$

Where  $p_i$  is the probability of a class  $i$  which could be a negative class or a positive class. The information gain is simply calculated by subtracting the entropy of  $Y$  given  $X$  from the entropy of  $Y$ , which is given an additional part of information  $X$  about  $Y$  as written below:

$$IG(Y, X) = E(Y) - E(Y|X) \quad (\text{eq. 2.2})$$

In many studies of fraud detection, DT is used as one of supervised learning models. A. Makolo et al. [29], E.A. Amusan [30] and P. Tiwari [31] use decision trees to build a model for credit card fraud detection and compare performances with other machine learning models. Although DT's models are not the best models with highest performance, the most advantage of using DT is to provide the highly interpretation of the model comparing with other ML models.

### III. Random Forest (RF)

Random forest is a supervised learning algorithm which manipulates many individual decision trees. Each DT casts a vote for a class prediction and the most voted class becomes the model's prediction as shown in Figure 2-5.

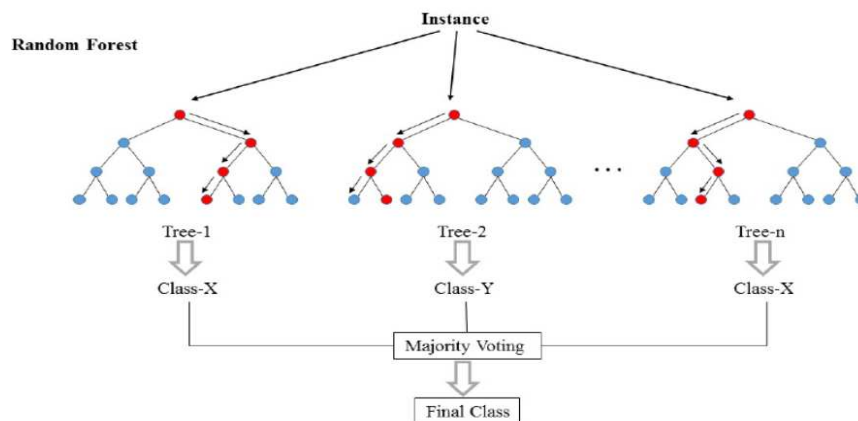


Figure 2-5. Random forest [8]

One of the disadvantages of using DT is overfitting and bias. DT keeps generating new nodes to fit the data very well including noise data, but it will become too complex tree which indicates that an accuracy of the DT model is very high with the train data but making many mistakes with new data. On that point, RF can mitigate overfitting by adopting bagging method, which is also known as bootstrap technique, that can build multiple DT models with different combinations of training data selected randomly. RF is a powerful and popular method in many studies for financial fraud detection.

C. Liu et al. [8] selected random forest as a financial fraud detection model. They used a RF model with a variety of combination of variables based on the feature importance measurement, feature selection and correlation analysis. The RF model could improve the detection accuracy efficiently. M. S. Kumar et al. [32] introduced a credit card fraud detection system using random forest algorithm. The accuracy of the detection system



was about 90%. R. Sailusha et al. [33] also used RF for credit card fraud detection in both online transactions and e-commerce systems. In order to verify the effect of the RF model, they used the Adaboost algorithm for comparison. The results of two models were measured by accuracy, recall, precision and F1-score. The RF model surpassed the Adaboost model in all results. However, the RF has the limit of taking long time when it deals with a vast number of trees. It is not effective for real-time predictions.

#### IV. Neural Networks (NNs)

Neural networks (NN), also known as artificial neural networks (ANNs), are series of algorithms that are inspired by the networks of biological neurons discovered in the human brain and consist of multiple node layers with threshold and weights, containing an input layer, one or more hidden layers and output layer. Each node is a perceptron which is the simplest NNs architectures and feeds the signal generated by multiple connections from nodes of the input layer to nodes of the output layer through the hidden layer(s) as shown in Figure2-6.

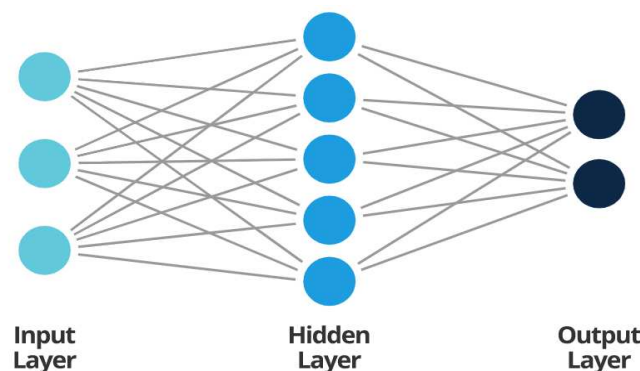


Figure 2-6. Artificial neural network architecture [91]

The depicted in Figure 2-7 below shows the fundamental function of neural network. Each input data ( $x_1, x_2 \dots, x_n$ ) is generated with its individual weight ( $w_1, w_2 \dots, w_n$ ) and

then these outputs are added ( $w_1 * x_1 + w_2 * x_2 + \dots$ ). The output of this activation function becomes  $y$ .

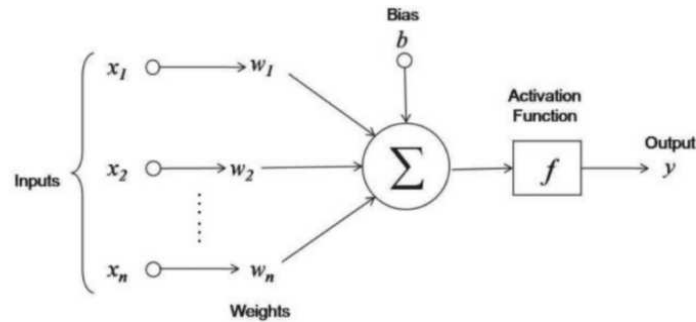


Figure 2-7. Activation function for NNs [35]

The most common formula known as binary threshold neuron becomes like:

$$z = b + \sum_i w_i x_i \quad (\text{eq. 2.3})$$

$$y = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{otherwise} \end{cases} \quad (\text{eq. 2.4})$$

NNs (ANNs) are applied in various studies in financial fraud detection as a fundamental idea.

R.Patidar et al. [35] have adopted NNs with genetic algorithm for credit card fraud detection that uses individual features in the input layer to train the card holder's traits and behaviour as shown in Figure 2-8. Genetic algorithm is used for determining the best number of hidden layers and nodes for credit card fraud detection. They adopted supervised learning feed forward back propagation method in NNs. When a new transaction arrives for approval, it is forwarded to a stream of authorization system that delivers the information for classifying fraud transaction or not. Performance of NNs combined with genetic algorithm was very efficient.

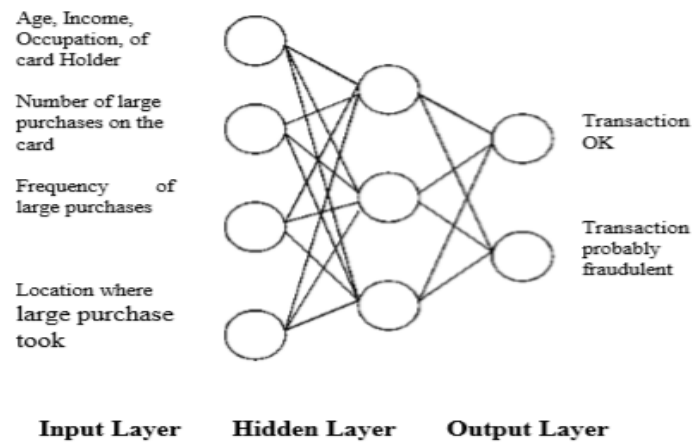


Figure 2-8. Layer of ANNs in credit card [36]

M. Kolali Khormuji et al. [36] used a cascade artificial neural networks for boosting recognition rate and accuracy of the fraud detection system. The cascade ANNs system aimed at achieving a very high recognition rate and reliability rate by gating networks that were utilised for congregating the confidence values of a few parallel artificial neural networks. In order to define the best weights for taking a balance between accuracy rate and reliability rate, the imperialist competitive algorithm (ICA) was used to achieve the whole optimal performance. ICA is a new progressive algorithm which has been proven in some other studies of having a good performance using imperialistic competition. In their experiments, the cascade ANNs with ICA led to the best performance to detect fraud transactions with about 98% accuracy.

There are a few disadvantages of neural networks. One of disadvantages is its black-box model. In comparison with a DT model, it is difficult to understand that why or how the NN models reached a certain output. Another disadvantage of using NN is that it takes much longer time to calculate input data than traditional ML algorithms because the neural network algorithm proceeds each feature to multiple connecting nodes in the

hidden layers. Therefore, it requires the quantity of computational power and sufficient memory for parallel processing and takes a high cost.

## 2.1.2. Unsupervised Learning

Unsupervised learning algorithm is suitable for classifying input data into some similar groups without using labelled data that is predefined tags like a fraud or not in the fraud detection case. It learns positions of each variable in input features and measures distances between them for clustering around similar groups to find the hidden data pattern from the given data. It has been used in a variety of studies in financial fraud detections for discovering fraud patterns by grouping customers in similar data patterns (Figure 2-9).

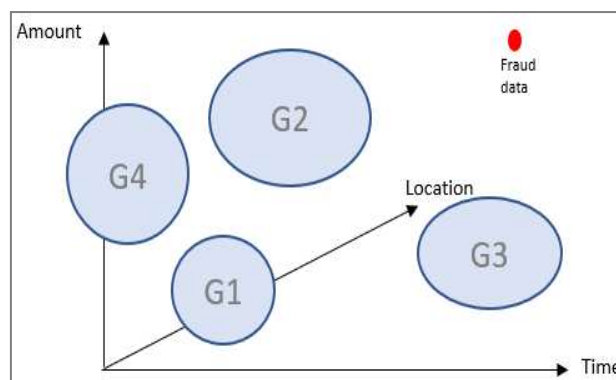


Figure 2-9. Image of grouping customers in similar data pattern

### I. Clustering (K-Means)

K-means clustering is the most popular unsupervised learning algorithm and is a centroid-based clustering algorithm that calculates the distance between a centroid data point and given data point. In order to discover underlying patterns, K-means inspects a settled number  $k$  of clusters in dataset. The number of  $k$  will be regarded as the number of centroids which is the centre of the cluster. After defining the number of  $k$ , k-means assigns all data points into the nearest cluster as shown in Figure 2-10.

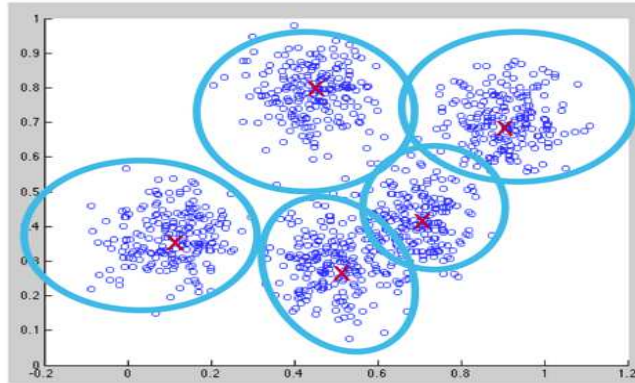


Figure 2-10. K-means clustering [92]

The processes of k-means algorithm are as below:

- i. Initialise the centre of the cluster by shuffling the dataset and randomly choosing  $k$  data points for specifying number of clusters  $k$ .
- ii. Calculate the sum of the squared distance between data points and all centre of data points (centroids).
- iii. Determine each data point to the centroid.
- iv. Estimate the centroids for the clusters by measuring the average of each data point that belongs to each cluster.

The objective of k-means is to minimize the squared error function. The objective function is as follows:

$$J = \sum_{i=1}^m \sum_{k=1}^K w_{ik} \|x^i - \mu_k\|^2 \quad (\text{eq. 2.5})$$

Where  $w_{ik} = 1$  for data point  $x^i$  if it exists in cluster  $k$ . If it does not belong to cluster  $k$ , then  $w_{ik} = 0$ .  $\mu_k$  is the centre point of  $x^i$ 's cluster.

In order to prevent customers from fraudulent online transaction, P. Singh [38] used the k-means method for clustering locations based on each IP address where customers have spent money via e-banking and the purposes of what they have spent money for.

M. Hegazy [39], B. Angelin [40] and B.A. Abdulsalami [41] also used the K-means clustering algorithm for discovering different behaviours between customers and fraudsters using credit cards by data mining.

## II. Isolation Forest (IF)

Isolation forest (IF) is an unsupervised learning algorithm for anomaly detection and performs “isolation” anomalies by building decision trees over features randomly. First two features are randomly selected and then, the data points are split by randomly choosing a value of the selected features between the minimum and the maximum. IF calculate anomaly score to determine how anomalous a data point is [94]. The algorithm uses the following anomaly score given a data point  $X$  and a sample size of  $n$ :

$$S(X, n) = 2^{-\frac{E(h(x))}{c(n)}} \quad (\text{eq. 2.6})$$

Where  $h(x)$  is the median exploration height for  $x$  from the isolation trees assembled, while  $c(n)$  is the median exploration depth to detect any normal node in the isolation trees.  $n$  is the number of external nodes which is located at the bottom of a tree in the sample size. The anomaly scores lie between 0 and 1.

When the observation score is close to 1, the path height (depth) is very short and then, the data point is simply isolated. It is judged as anomaly. When the observation score is smaller than 0.5, the path height (depth) is long, and then, it is judged as a normal data point as illustrated in Figure 2-11.

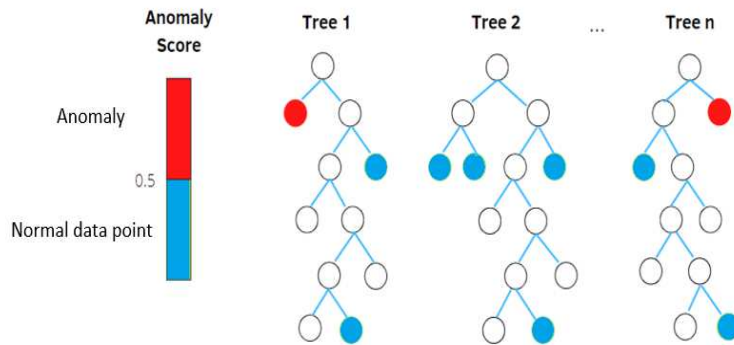


Figure 2-11. Anomaly detection with isolation forest [94]

The scores for each tree are calculated and the averages cross different trees obtain the final anomaly score for whole forest. The closest anomaly score is 1. Figures 2-12 and 2-13 show the sub dataset that was split by random tree choice based on the isolated data point to create a forest. Figure 2-12 describes the example of isolating a non-anomalous point whereas Figure 2-13 shows the example of isolating an anomalous point.

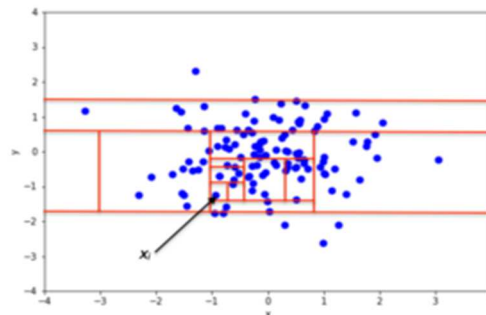


Figure 2-12. Example of isolating a non-anomalous point in a 2D gaussian distribution [41]

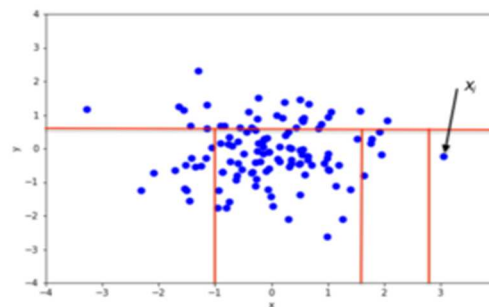


Figure 2-13. Example of isolating an anomalous point in a 2D gaussian distribution [41]

There are many studies for credit card fraud detection using IF algorithm [42] [43] [44] [45] and they demonstrated and achieved a high accuracy of the fraud detection systems using IF algorithm.

S. S. Negi et al. [45] proposed to use IF and local outlier factor (LOF) algorithms. They showed higher effectiveness of using these algorithms than other machine learning algorithms such as random forest, naïve bays, and support vector machine. The accuracy of the IF model was approximately 99% which was the best score among other model's ones. In the several studies for credit card fraud detection [42] [43] [44], they also proposed to use IF algorithms which is good at detecting anomaly samples and will achieve a great accuracy.

### III. Local Outlier Factor (LOF)

Local outlier factor is an unsupervised learning algorithm for anomaly detection and calculates the local density deviation of an input data point regarding its neighbours. Outliers are described based on a concept of a local density. The local density is provided by k-nearest neighbours, whose distance is used to evaluate the density. Outliers are points that have a substantially lower frequency than their neighbours. The number of neighbours is typically set larger than the minimum number of cases a cluster must contain, thus other samples can be local outliers relating to the cluster.

All local reachable densities of each point are calculated as below [46]:

$$lrd\ k(o) = \frac{\|Nk(o)\|}{\sum_{o' \in Nk(o)} reachdist\ k(o' \leftarrow o)} \quad (\text{eq. 2.7})$$

$\|Nk(o)\|$  indicates number of neighbours. Refer to the following equation, where  $o$  is the point in the centre and  $o'$  is a point near it.

$$reachdist\ k(o \leftarrow o') = \max\{dist\ k(o), dist(o, o')\} \quad (\text{eq. 2.8})$$



To calculate the LOF score for a specific point, first, the  $k$ -nearest neighbours  $\|Nk(o)\|$  should be determined for each data point. Second, the local density for a data point is estimated by calculating  $lrd\ k(o)$  by using the  $\|Nk(o)\|$ . Then, the LOF score is calculated by the following equation:

$$LOF\ k(o) = \frac{\sum_{o' \in Nk(o)} \frac{lrd\ k(o')}{\|Nk(o)\|}}{\|Nk(o)\|} \quad (\text{eq. 2.9})$$

$$= \sum_{o' \in Nk(o)} lrd\ k(o') * \sum_{o' \in Nk(o)} reachdist\ k(o' \leftarrow o) \quad (\text{eq. 2.10})$$

The Figure 2-14 below describes the local outlier factor with the minimum number of set 3 nearest neighbours, which is referred to as Min Pts.

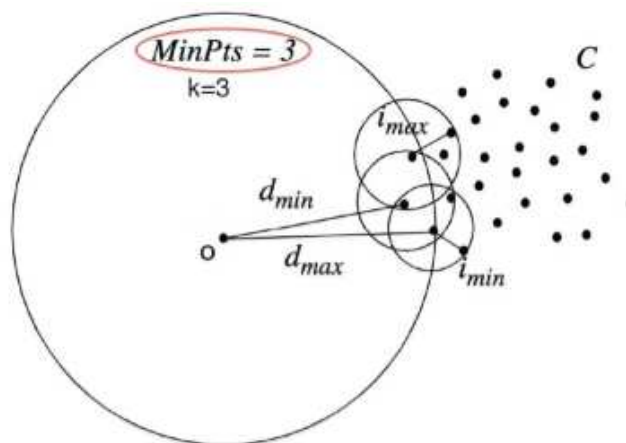


Figure 2-14. The logic of local outlier factor [46]

This algorithm is also popularly used in several studies for credit card fraud detection [46] [47] [48].

S. Jaiswal et al. [47] and H. John al. [46] also suggested to use both algorithms of IF and LOF for credit card fraud detection because the credit card datasets are highly skewed and imbalance. These unsupervised learning algorithms evaluate the different data point between a fraud and customer by calculating the local deviation of the density of the input

case in relation to its neighbours. In their experiment, the LOF model achieved highest accuracy rate of 97% followed by the IF (76%).

D. Tripathi et al. [48] suggested to use LOF for credit card fraud detection. They evaluated the performance over the different nearest neighbours regarding the rates of true negative and false negative, accuracy of the detection system. In their experiment, they used the two types of datasets that are both fraudulent transaction dataset and imbalance target data. Dataset 1 contains 100,000 transactions with 2,659 fraudulent actions, which is 100:3 ratio of the imbalance transaction cases, published in UCSD-FLCO competition. Dataset 2 has 94,682 transactions with 2,094 fraudulent transactions, which is 9:3 ratio, provided by the University of California, San Diego. The LOF model was built with these datasets and different K-nearest neighbours. The results showed that the accuracy of each model is lying between 60 and 69% for dataset 1 and 96% for dataset 2 with alternative neighbours. The larger volumes of transactions were precisely detected by the LOF model.

## **2.3. Deep Learning**

Deep learning is a subfield of machine learning algorithms and uses a structure of multiple layers based on neural networks shown in Figure 2-15. It is designed and works like a human brain. The layers in deep learning can learn implied representation of input raw data without feature extraction that focuses on reducing the number of features from the original set by creating new summarised original features set.

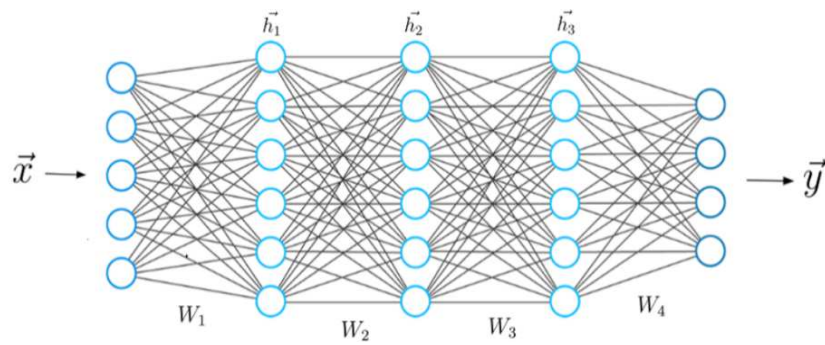


Figure 2-15. Deep learning neural networks [49]

The step of feature extraction is contained in the neural network layers. As described in the content of NNs previously, the typical NNs is composed of several layers, input layer, two or more hidden layers and output layer. The concept of deep learning is fundamentally same as NNs.

Deep learning has been popularly used for image, audio and video recognitions in terms of coping with big data in depth. It learns input data by dividing it into a plurality of segmented data patterns through many hidden layers. Recently it came to be used for classification issues such as fraud detection in financial area.

## I. Autoencoder Neural Networks (AE)

Autoencoder is an unsupervised deep learning algorithm [49] and a type of ANNs as shown in Figure 2-16. It learns how to compress and decompress input data for representation of the original input data and discovers specific features from the given data during the process of data compression, also known as dimensionality reduction, and how to map the compressed features to the latent layer. The autoencoder finds out how to reconstruct the input data from mapping the features. The most advantage of using autoencoder for financial fraud detection is that autoencoder does not need fraudulent transaction data to learn fraud patterns.

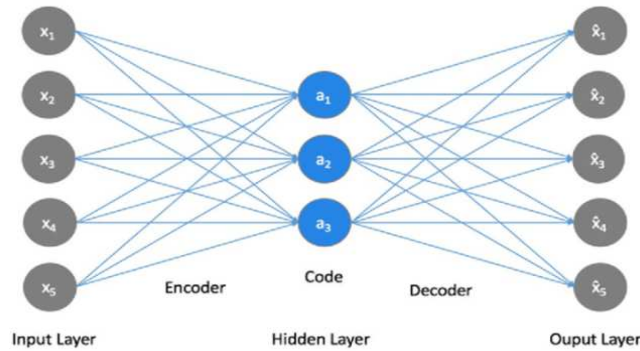


Figure 2-16. Autoencoder with hidden layers [49]

In order to measure how well the input data can be reconstructed, a loss function is calculated for updating different weights and reducing the loss between the represented data and the original data. Autoencoder uses unlabelled training data  $\{x(1), x(2), x(3), \dots\}$ , where  $x(i) \in \mathbb{R}^n$  and applies backpropagation to learn how to approximate to a function  $h_{w,b}(x) \approx \hat{x}$  as displayed in Figure 2-17. The output  $\hat{x}$  is similar to  $x$ .

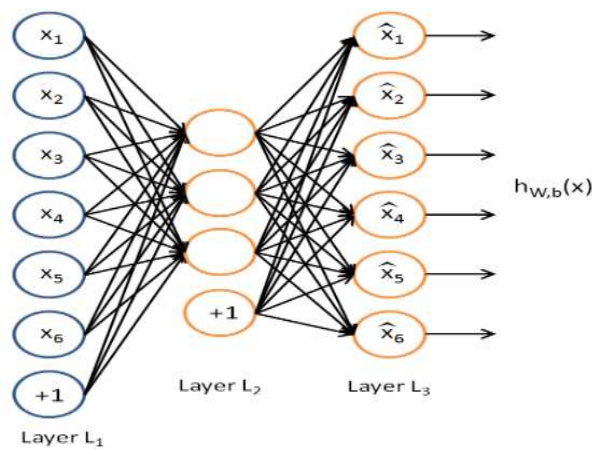


Figure 2-17. Autoencoder neural networks [49]

There are three main layers of autoencoder: encoder, hidden and decode.

### (a) Encoder Layer

An autoencoder model learns how to reduce dimensions of input features and compress the given data into an encoded representation.

### (b) Hidden Layer

This layer holds the compressed representation of the given data and expresses the most compacted dimensional features.

### (c) Decoder Layer

The model learns how to reconstruct the compressed data to the original data by using the loss function and calculates the loss between the original data and the reconstructed data.

The loss function  $l(x, \hat{x})$  is measured by gradient descent over the parameters of encoder and decoder networks [49]:

$$l(x, \hat{x}) = \|x - \hat{x}\|^2 = \|x - h_{w,b}(x)\|^2 \quad (\text{eq.2.11})$$

The equation of encoder and decoder are given as follows [49]:

$$\text{Encoder} \quad h(x) = \sum(Wx) \text{ or } \text{Tanh}(Wx) \quad (\text{eq.2.12})$$

$$\text{Decoder} \quad h_{w,b}(x) = \sum(W*h(x)) \text{ or } \text{Tanh}(W*h(x)) \quad (\text{eq.2.13})$$

The proportion of fraud transaction data is very little whereas the number of legitimate transaction data is very large. It is difficult to keep track of new fraudulent behaviour and state-of-the-art fraud schemes from a few fraud samples because fraudulent actions are not carried out by one person. On the other hand, legitimate transactions are carried out by the same customer who holds his or her own bank account or credit card. Autoencoder can reconstruct customers' behaviour patterns by learning from specific features among large history transaction data. Autoencoder models judge fraudulent data by using loss function with mean squared error (MSE) that measures the error distance of variables in specific features between the learnt data and new input data. There are some related

studies of fraud detection using the autoencoder model [49] [50] [51] [52] and they chose autoencoder techniques from the perspective of coping with unbalanced transaction datasets. They commonly use two popular techniques of feature engineering, which are principal component analysis (PCA) and standardisation.

PCA is a technique of dimensionality reduction and uses orthogonal transformation that computes covariance matrix which represents the correlation between two variables. Unlike machine learning models, deep learning is essential for data processing standardisation as it standardises and weight each attribute to measure how much specific features influence.

Standardisation is an essential data processing for using deep learning because deep learning multiplies each attribute and sets the weighting coefficients. Deep learning has not implemented feature engineering on input data from the point of view of adding latent data patterns.

Fraud transaction data is always imbalanced and needs to be carefully handled while using machine learning algorithms. Popular methods of coping with imbalanced datasets are oversampling and under sampling which are techniques to balance the class distribution. Oversampling is utilised to synthesise new samples of fraudulent classes but, it will take in noise. Under sampling removes samples from the majority class in the trained dataset but, it may remove useful information or important data.

Autoencoder is good for coping with imbalanced datasets without considering the minority class issue because it only uses the majority class samples. Some researches for credit card fraud detection use an autoencoder model [50] [51] [52]. P.Jiang et al. [50] designed a six-layer autoencoder for the dataset and selected SoftMax with cross-entropy

as the loss function for final classification to detect credit card fraud. The autoencoder model improved the classification accuracy of the fraud class when the threshold was equal to 0.6 rather than the other rate of thresholds. A. Pumsirirat et al. [52] used deep learning based on auto-encoder and restricted Boltzmann machine, also called stochastic Hopfield network with hidden units, for credit card fraud detection. Fraudsters gain new technology that enables them to steal money from customers. Their autoencoder applied backpropagation by setting the input data equal to the output data. Restricted Boltzmann machine is a set of random quantities having the memoryless property of a stochastic process and it can reconstruct legitimate transactions to discover fraudsters from legitimate patterns and holds two layers, input layer and hidden layer. They used the library of TensorFlow to implement autoencoder and restricted Boltzmann machine. The number of studies of financial fraud detection using autoencoder is not a few, but almost all studies use only raw data as the input data for autoencoder. They do not apply feature engineering methods to the raw data because DL methods have a function of feature extraction to reduce the number of features in an input data and automatically learns features at multiple levels by combining the input features.

## **II. Convolutional neural networks (CNNs)**

The aim of CNNs is to find patterns in image features to recognise objects and classes via convolutions where the input data filters the information and produces a feature map. CNNs is well suited to image classification and recognition and efficiently used for helping to build a more robust feature space based on a signal in Figure 2-18.

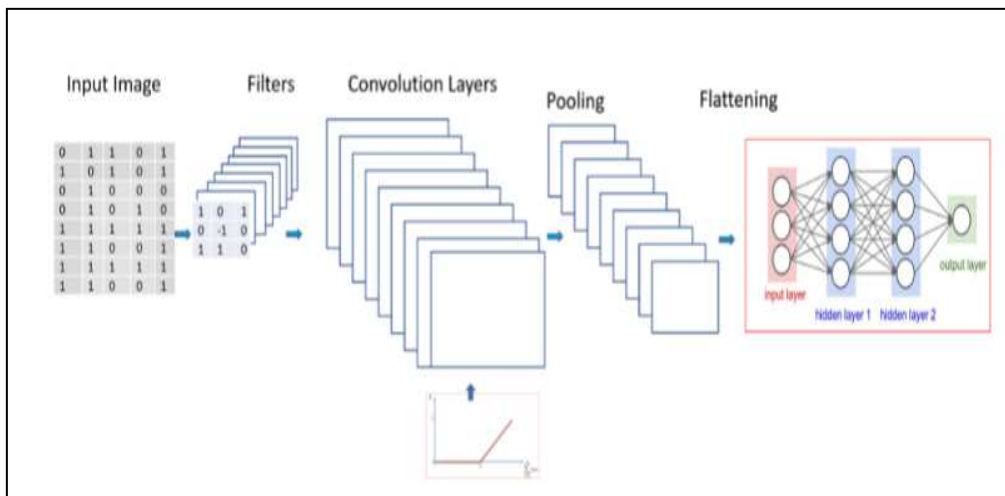


Figure 2-18. Convolutional neural networks architecture [54]

CNNs transform the input data from the input layer through all connected layers into a set of class scores provided by the output layer. Although there are a variety of CNN architectures, they are based on the pattern of layers as shown in Figure 2-19. There are three main groups: input layer, feature extraction layers, and classification layers. The input layer accepts three-dimensional input commonly in the form of the size of the image and has a depth representation of the color channels. The feature extraction layers consist of a general repeating pattern of the sequence: convolution layer and pooling layer. Convolutional layers have parameters for the layer and additional hyperparameters. Pooling layers are generally inserted between successive convolutional layers and reduce the data representation progressively over the network. They operate independently on every depth slice of the input. These layers discover several features in the images and progressively construct higher-order features. This corresponds directly to the continuing theme in deep learning by which features are automatically learned as opposed to traditionally hand-engineered. Lastly, classification layers in which one or more fully connected layers take the higher-order features and produce class probabilities.



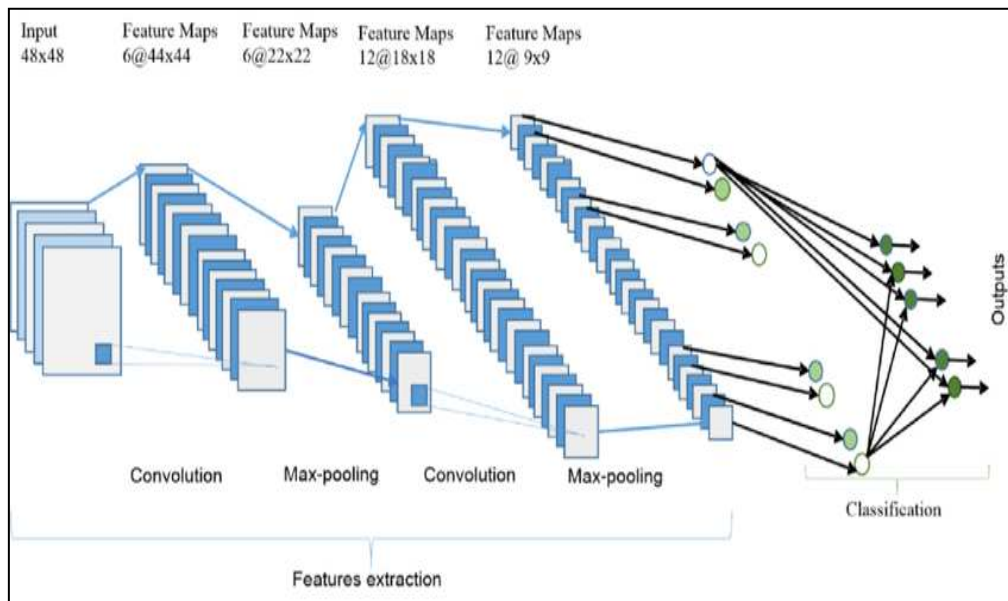


Figure 2-19. Basic structure of CNNs [54]

CNNs are applied in many cases of dealing with MRI data [55], 3D shape data [56] and graph data [56]. Even though the main purpose of using CNNs in a variety of studies is to handle the specific image and audio data, there are seldom cases using CNNs for classification such as financial fraud detection.

Z. Zhang et.al [57] proposes a fraud detection model using the convolutional neural network in the online transaction field. In the paper, the CNN model constructed an input feature sequencing layer that carries out the reorganisation of raw transaction features to assemble different convolutional patterns. The experimental data was provided by a commercial bank, and it was a total of about five million data of a six-month sequential transaction data. They used the sequential transaction data only and processed the data to multiple dimensions for using CNNs model. Their purpose was not to learn fraud behaviour but tried to reconstruct the input by handling the processed transaction data as well as image or audio records.

### III. Recurrent neural networks (RNNs)

RNNs means chaining multiple layers to create a sequence of dependent computations and are influenced by what it has learnt from the past which is also called as memories. When RNNs learn input data, they recall things learnt from prior inputs while producing outputs. A different output could be produced with the same input based on previous inputs in the series. As shown in Figure 2-20,  $x_1, x_2, x_3$  are inputs where  $h_1, h_2, h_3$  indicate hidden layers. RNNs use a loss function to calculate the outputs ( $y_1, y_2, y_3$ ). The loss is later backpropagated and weights ( $W$ ) are updated.

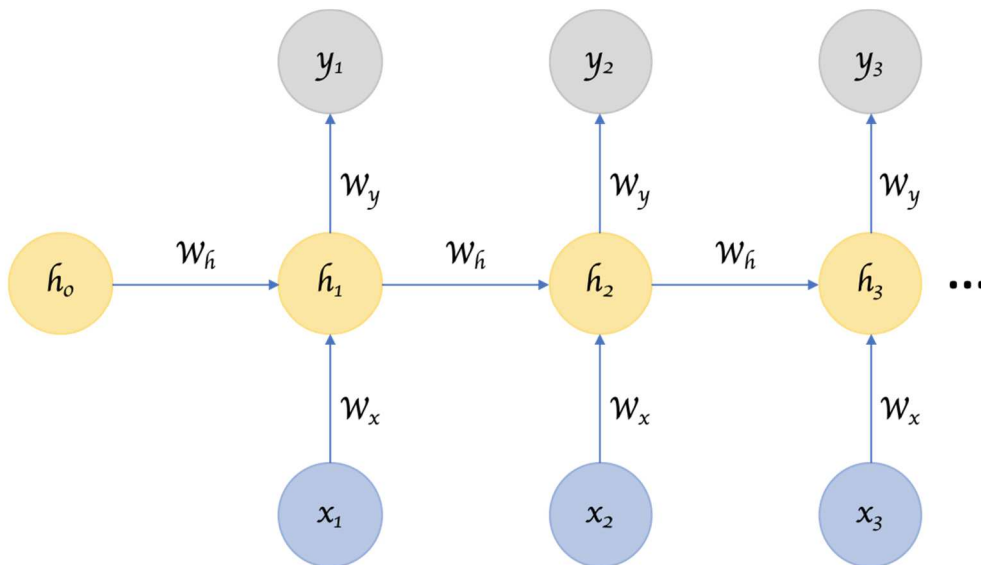


Figure 2-20. Recurrent neural networks with a hidden state [94]

The network in RNNs may have many hidden states and the same activation function in each hidden state is computed and produces the output of each layer. The activation function is defined as below [94]:

$$h^{(t)} = f(h^{(t-1)}, x^{(t)}, \theta) \quad (\text{eq.2.14})$$

Where activation function  $f()$ ,  $h^{(t-1)}$ : previous step,  $x^{(t)}$ : input,  $\theta$ : parameters

RNNs can have loops in the connections and model temporal behaviour gain accuracy in domains. RNNs is suitable for dealing with sequential data such as time-series prediction, video analysis, and music information retrieval. RNNs are a superset of feed-forward neural networks but they add the concept of recurrent connections. The connections span adjustment time-steps such as a previous time-step, giving the model the concept of time.

Although the conventional connections do not have cycles in recurrent neural networks, recurrent connections can form cycles including connections back to the original neurons themselves at feature time-steps as displayed in Figure 2-21.

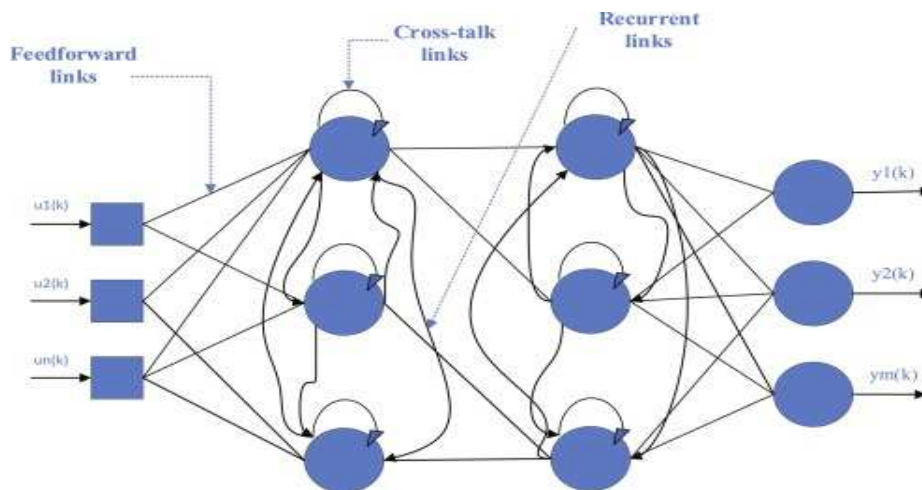


Figure 2-21. RNNs architecture [59]

The output is calculated from the hidden state at the given time-step. The previous input vector at the previous time step will influence the ongoing output at the ongoing time-step through the recurrent links.

Long Short-Term Memory (LSTM) networks are the most general used variation of RNNs, which were introduced in 1997 by Hochreiter and Schmidhuber [59] and remember values over arbitrary intervals. The crucial component of the LSTM is the memory cell and both gates: the forget gate and the input gate. The contents of memory

cell are modulated by the input gates and forget gates. The gating structure allows information to be retained across many time-steps and consequently allows gradients to flow across many time-steps. LSTM networks consist of many connected LSTM cells and perform well in how efficient they are during learning. In the concept of the feed-forward multilayer neural networks with RNNs, each node connects the output of a hidden layer neuron as an input to the same hidden layer neuron as shown in Figure 2-22.

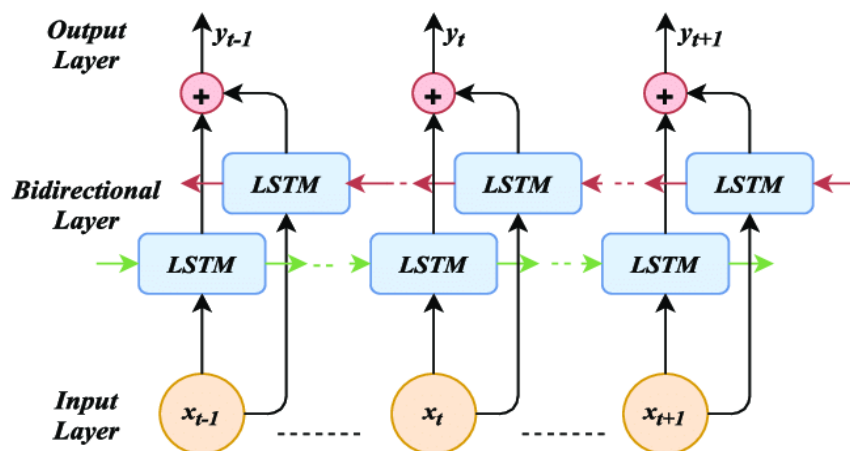


Figure 2-22. The concepts of RNNs with LSTM [62]

Each LSTM unit has two types of connections:

- Connections from the previous time-step (outputs of those units)
- Connections from the previous layer

The memory cell in an LSTM network is the central concept that allows the network to maintain state over time. The basic layer accepts an input vector  $\mathbf{x}$  and gives output  $\mathbf{y}$ . The output  $\mathbf{y}$  is influenced by the input  $\mathbf{x}$  and the history of all inputs. The layer is influenced by the history of inputs through the recurrent connections. The RNN has some internal state that is updated every time when a vector is inputted to the layer [60].

The data scientist team at Barclays which is a British multinational universal bank has released the paper about RNNs for fraud detection on debit card transactions [61]. According to the paper, they used deep recurrent neural networks on debit card transactions and compared the detection performance with classical ML approaches which they have already introduced in the fraud detection system. Using the RNN model with LSTM on their own a large transaction data had better performance than others.

I. Benchaji et al. [62] also developed a credit card fraud detection system using LSTM networks as a sequence learner including transaction sequences. The purpose of the study was to capture the historic purchase behaviour of credit card users with the goal of improving accuracy of a fraud detection model on new incoming transactions. The dataset they used was generated by a multi-agent-based simulation methodology based on a sample of aggregated real transaction data from a private Spain bank and it contained transactions corresponding to card purchases occurred for 180 days and consisted of almost 600,000 transaction records with 7,200 fraudulent records. The LSTM model's performance was measured by using AUC and the Mean Square Error over the last 10 epochs and show quite high accuracy.

## 2.4. Feature Engineering

### 2.4.1. Feature Aggregation for Evolving Customer Behaviour

In many studies of feature engineering in financial fraud detection, feature aggregation is the most popular method to evolve a customer's behaviour when credit card's transaction occurs. When a transaction is carried out, it links with some features such as time, date and amount. The fundamental concept of feature aggregation is to create a new feature by aggregating these individual feature based on customer behaviour. Aggregation makes more detailed features that express customer's own transactional patterns related to for example to geo-location and the amount of money and the time stamp. Based on customer's ID, some action features are aggregated as the image shown in Figure 2-23.

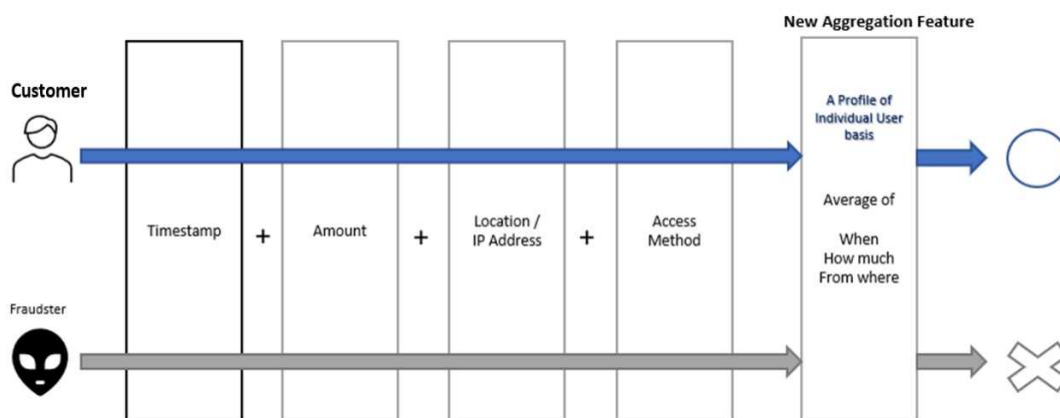


Figure 2-23. Feature aggregation for evolving customer behaviour

For instance, aggregated features can be “the average amount by transaction device per day in the past one week”, “number of transactions via a specific IP address per day in the past two weeks”, “the average amount per day over the past one week” and “the average amount spent per day over the past 15 days”.

Many studies of feature engineering for financial fraud detection have used feature aggregation methods to create new features that expose customer’s behavioural patterns on transaction [28] [65] [66] [67].

Zhang et al. [66] suggested a new feature engineering methodology that employs homogeneity-oriented behaviour analysis (HOBA) which generates feature variables by using a feature aggregation method based on recency, frequency and monetary (RFM) and groups into homogenous fraudulent patterns for credit card fraud detection. The RFM is popularly used for behaviour analysis in the marketing area. For instance, “How recently did a credit card holder make a transaction”, “How often did a credit card holder make transactions”, “How much did a credit card holder spend in transactions” and “Where did a credit card holder make transactions”. These factors were created by feature aggregation related to user’s behaviour analysis based on regular intervals during a transaction as seen in Figure 2-24. To evaluate the effectiveness of applying the new methodology, they used SVM, RF,CNN and RNN for the experiment. The performances of the models built with the HOBA features exceeded the performances of the models built without the HOBA features in all measures.

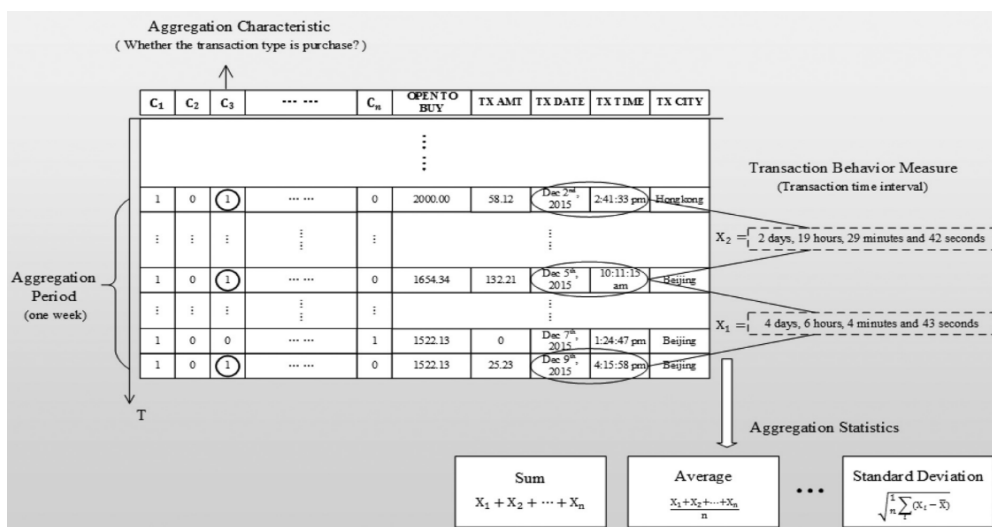


Figure 2-24. Applying the transaction aggregation process with the HOBA principle [66]

The methodology of using HOBA principle was used for only time stamps in their study. However, it may not be sufficient to create aggregated features that reveal different behaviour between a customer and fraud with only a time stamp attribute. Furthermore, the HOBA principle does not contain a feature selection concept.

Y.Lucas et al. [65] suggested using a feature aggregation framework based on multi-perspective Hidden Markov Models (HMMs) for credit card fraud detection. The history of credit card transactions has the card holder’s habits of the timing or the place of using a credit card in the last 24h. HMM is a sequence classification model which considers the sequential properties of transaction data. The multi-perspective HMMs categorise a symbol on transactions such as “merchant and amount”, “timing”, “fraud or customer”, “genuine” and observe each symbol as the sequential event on transactions. The HMMs calculate the likelihood of sequences of observed symbols and create features of each event as shown in Table 2-1. To measure the effectiveness of the addition of the HMM features, they use perspective, recall and AUC metrics, and random forest as an experimental model. Consequently, the use of the HMM-based features improved the precision-recall AUC of the random forest model significantly compared with the use of the original features only.

Feature	Signification
AGGCH1	# transactions issued by user in 24h.
AGGCH2	Amount spent by user in 24h.
AGGCH3	# transactions in the country in 24h.
AGGCH4	Amount spent in the country in 24h.
AGGTM1	# transactions in terminal in 24h.
AGGTM2	Amount spent in terminal in 24h.
AGGTM3	# transactions with this card type in 24h.
AGGTM4	Amount spent with this card type in 24h.

Table 2-1. Aggregated feature creation on the card holders and the terminal [65]



A. Bahnsen et al. [67] deployed the transaction aggregation strategy and suggested to create a new feature aggregation set for monitoring the spending customer's behaviour patterns based on evaluating the periodic behaviour of the transaction time using the method of von Mises distribution. They used the aggregation strategy on real credit card transaction data provided by a large European card institution and created time features by performing calculations such as the number of transactions in the last 24 hours, the sum of the transaction's amounts in the same time period and so on. Table 2-2 provides calculation example of aggregated features. Where  $x_i^{a1}$  is the number of transactions in the last 24 hours,  $x_i^{a2}$  is the sum of the transaction's amounts in the same time period,  $x_i^{a3}$  is the number of transactions with the same transaction type and same location in the last 24 hours and  $x_i^{a4}$  is the sum of the transactions amounts of the transactions with the same type and location in the last 24 hours.

Raw features						Aggregated features			
Trxld	Cardld	Time	Type	Country	Amount	$x_i^{a1}$	$x_i^{a2}$	$x_i^{a3}$	$x_i^{a4}$
1	1	01/01/15 18:20	POS	Luxembourg	250	0	0	0	0
2	1	01/01/15 20:35	POS	Luxembourg	400	1	250	1	250
3	1	01/01/15 22:30	ATM	Luxembourg	250	2	650	0	0
4	1	02/01/15 00:50	POS	Germany	50	3	900	0	0
5	1	02/01/15 19:18	POS	Germany	100	3	700	1	50
6	1	02/01/15 23:45	POS	Germany	150	2	150	2	150
7	1	03/01/15 00:00	POS	Luxembourg	10	3	400	0	0

Table 2-2. Example of calculation of aggregated features [67]

As a result, they showed the effectiveness of using the aggregated features for a fraud detection model by comparing the performance of machine learning models with/without the aggregated features.

Both studies of Y. Lucas et al. [65] and A. Bahnsen et al. [67] have demonstrated the impact of using feature engineering methods on data with improved performance of machine learning models. In their works, they focused only on the feature aggregations

side to reveal latent fraudulent patterns. Transaction data in financial institutions has many common attributes such as time, amount, balance, deposit, credit history, bank information, access devices and so on. In the studies of using feature aggregation for transaction data, they create quite similar patterns of features based on the user behaviour on transactions specifically using time and amount. On the other hand, it may be vulnerable to intrusion from advanced fraudulent schemes by only implementing feature aggregation methods on the features pertaining to transaction.

### **2.4.2. Feature Transformation using Mathematical Equations**

There are several mathematical equations for transforming a single attribute into other dimensions by mapping data. The most popular techniques of feature transformation are counts, subtraction, multiplication, deviation, average, maximum, median, minimum, standardisation, and logarithm transformation and they are introduced in some research of feature engineering for improving machine learning performances by using mathematical functions for classification problems [68] [69] [70] [71].

A.Nagaraja et al. [68] introduced an approach for any network anomaly detection using feature transformation based on mathematical methods. They used feature clustering based on the gaussian distribution function and a k-Nearest Neighbours (KNN) classifier as a detection model for finding the similarity between observations. The distribution function provides the equivalent deviation and threshold values to carry similarity calculation, and then the distance function of KNN measures the distance of the transformation features and determines if the input is fraud or a legitimate value. Using transformation features improves the detection accuracy in comparison with using the raw data only. J. Heaton [69] proved that performances of the studied classification

models with various types of transformed features were improved in comparison with performances of the models with only raw features. In his study, he created new features by using the sixteen methods of feature transformations such as counts, rational differences, polynomials, distance formula, distance between quadratic roots, powers and logarithms. Then, he selected the four model types of deep neural networks, gradient boosted machines, random forests and support vector machine and evaluated the model's performances. F. Nargesian et al. [70] suggested to use Learning Feature Engineering (LEF) which is a tool for automatically determining the effective features from the performances based on the examinations of their mathematical transformations such as log, square roots, round, sigmoid, subtraction, tanh, and the other four arithmetic calculations. The LFE method is utilised with a multi-layer perceptron classifiers (MLP) as the automated learning algorithm. It learns from each pattern of performances based on the mathematical transformation data and then, selects the best features based on the correlation coefficient as a new feature set. The result of comparison of the machine learning (ML) models with using engineered features and non-engineered features shows that the ML models with the engineered features improved a rate of classification accuracy rather than using the independent original raw features. K. Veeramachaneni et al. [71] introduced feature transformation tools and technique of using mathematical equations such as average, sum, standardisation, Min-Max normalisation, and log transformation. They examined created features by using feature transformation tools on various classification issues, i.e., analysing online behaviour, health condition related social networks and fraud detection, and presented the possibility of building classifiers more effectively by using the transformed features set.

The all studies of A. Nagaraja et al. [68] , J. Heaton [69], F. Nargesian et al. [70], and K. Veeramachaneni et al. [71] have demonstrated that the effectiveness of using feature transformation methods. However, they have not considered feature creations based on analysing human’s behaviour.

Feature transformation is primarily used for dealing with data of images and audio records [15] [16], i.e., picture elements (pixel), MPEG-1 Audio Layer-3 (MP3) and Moving Picture Experts Group (MPEG) and these datasets are used by deep learning techniques for image or voice recognition. As explained about a deep learning structure in Section 2.3., deep learning processes input features by encoding and decoding and creates new features through the multiple hidden layers and learns hierarchical feature representations as shown in Figure 2-25 [72].

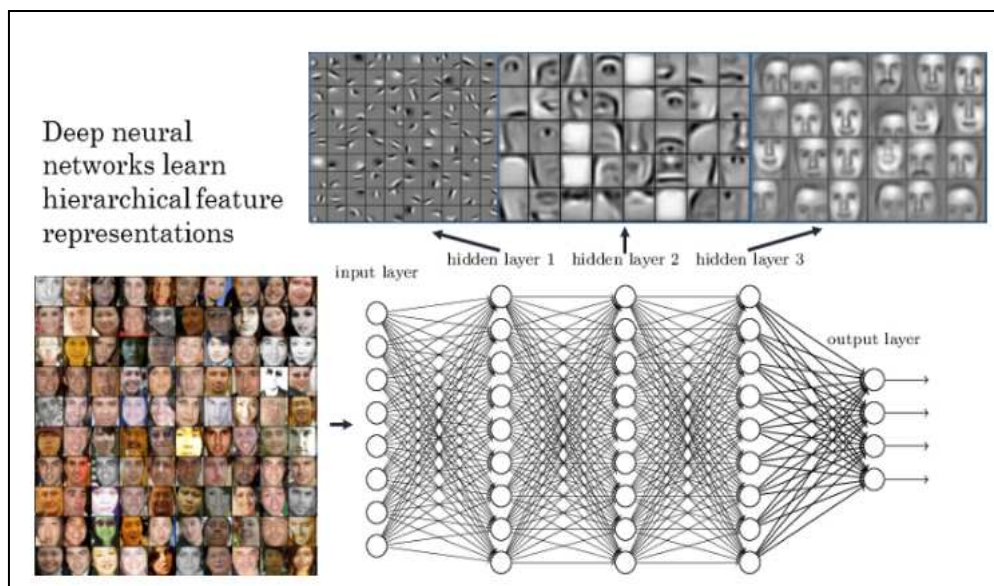


Figure 2-25. Example of how to deal with the image recognition data in deep learning processes [72]

Each layer can have a different number of neurons and each layer is fully linked to the next layer. Figure 2-26 shows the three components of  $input_0$ ,  $input_1$ , and  $input_2$ , which connects to the next node by using weights and biases accordingly. New transformed

features are created by combining each input with weights in the next layer, which is known as feature engineering in deep learning.

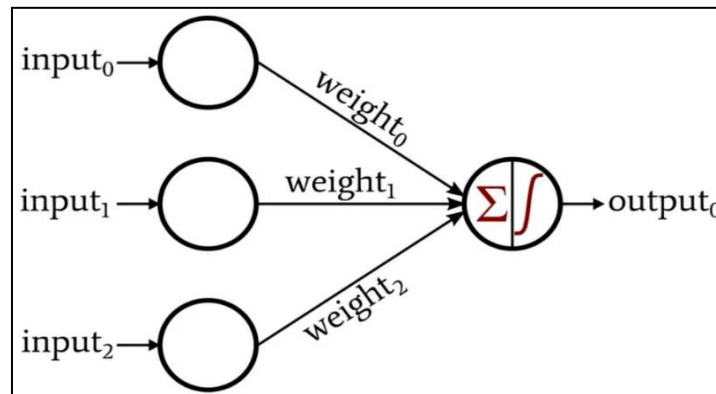


Figure 2-26. Connections between the nodes with each weight [95]

For deep learning algorithms, feature transformation is to set evaluation criteria for values in features and is necessary because the input features are calculated together and measured equally. In order to arrange standards for features, feature scaling is required before creating a deep learning model.

### A) Feature Scaling

In each feature, data values will be in a wide range of numbers and various standards. Feature scaling is a way where the range of the data variables in independent features is normalised, and it will support all various standards of a wide range of variables in the same range. There are two popular methods of feature scaling: Standardisation and Normalisation. Normalisation, also known as min-max scaling, is better used for the data variables in each feature which does not follow a gaussian distribution and good to use for algorithms which do not expect any distribution of the data such as a neural networks and K-nearest neighbours. Normalisation is a necessary pre-processing procedure in Neural Networks algorithm that demands data variables in all features on a 0 to 1 scale. However, it will be given highly impact on by outliers.

In contrast, standardisation is good to use for algorithms which expect to use the data which follows a gaussian distribution. Standardisation will be not affected even though the data variables include outliers in the feature because it does not have a bordering range. Figure 2-27 shows an instance of the independent features of alcohol and Malic Acid content in the wine dataset provided by the UCI machine learning repository [73]. We can see the impact of the two feature scaling techniques of normalisation and standardisation on the data.

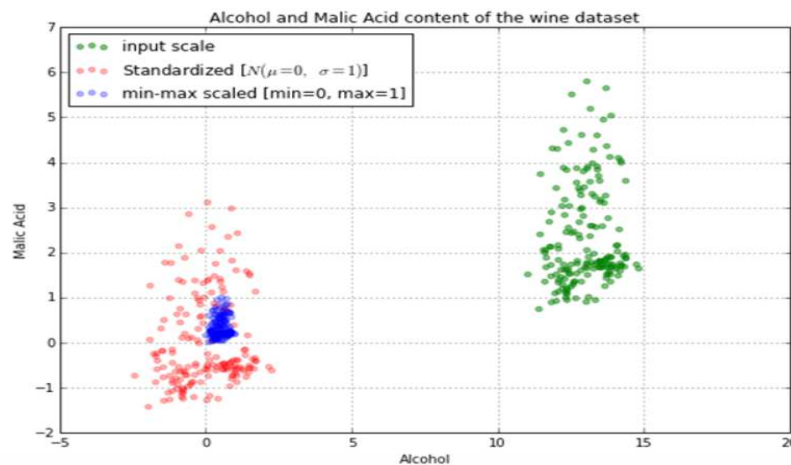


Figure 2-27. The impact of feature scaling on the wine dataset [73]

The plot in Figure 2-27 describes three different scales: the green dots are volume-percent, the red dots are standardised features, and the blue dots are normalised features.

- **Normalisation**

The Min-Max normalisation transforms  $x$  to  $x'$  by rescaling the range of data variables in features to scale the range in  $[0,1]$ . The formula for Min-Max normalisation  $x'$  (normalised value) is defined as below [17]:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \quad (\text{eq. 2.15})$$

Where:  $x$  means original value,  $\max(x)$  and  $\min(x)$  indicate the maximum and the minimum values of the feature appropriately.

- **Standardisation**

Standardisation is a process of adjusting the data values for obtaining the characteristics of standard normal distribution [17]. The data values standard in each feature are rescaled by calculating the following formula [17]:

$$\text{Standardisation} = \frac{x - \text{mean}(X)}{\text{Standard Deviation}(X)} \quad (\text{eq. 2.16})$$

Mean ( $\bar{X}$ ) is computed by only using the values in the independent feature. Standard deviation is a method of scaling the values based on z-score which calculates the following equation [17]:

$$\text{Standard Deviation} = \sqrt{\frac{\sum_{i=1}^n (X_i - \bar{X})^2}{N-1}} \quad (\text{eq. 2.17})$$

Where:

$X_i$  = Value of each data point

$\bar{X}$  = Mean

N = Number

## **B) Principal Component Analysis (PCA)**

Principal component analysis (PCA) is a technique for reducing feature dimensions from the original feature dimensions but keeps the meaningful variation in the original attributions. PCA explores correlations among the given data and produces new aggregate variables which is a condensed dimensional feature, called principal components (PC).

For instance, Figure 2-28 describes features of three dimensions in original data space mapped to two dimensions in component space by applying PCA [75]. There are practically three fundamental schemes of using PCA. First, it is helpful to analyse correlated variables. Second, it is good to reduce redundant features that are unrelated to a target. Lastly, it is useful to divide mixed up data patterns in original data space into classifying each pattern in component space where machine learning algorithms can easily differentiate these patterns [76].

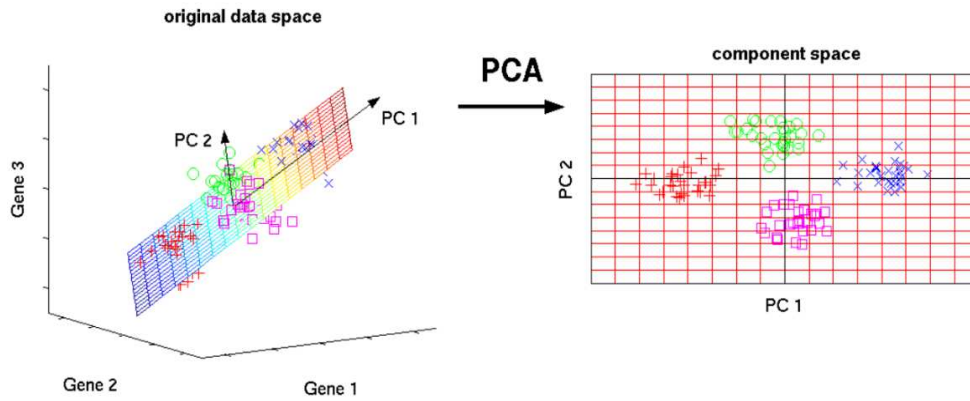


Figure 2-28. Feature transformation using PCA [75]

The PCA formula is given the following [75]:

Samples  $X_1, X_2, \dots, X_N \in \mathbf{R}^n$  of the variable  $X \in \mathbf{R}^n$  that randomly selected.

$$\max_{\|a\|=1} \frac{1}{N} \sum_{i=1}^N (a^T (X^i - \frac{1}{N} \sum_{j=1}^N X^j))^2 = \max_{\|a\|=1} a^T V_{xx} a \quad (\text{eq. 2.18})$$

$$V_{xx} = \frac{1}{N} \sum_{i=1}^N (X^i - \frac{1}{N} \sum_{j=1}^N X^j) (X^i - \frac{1}{N} \sum_{j=1}^N X^j)^T \quad (\text{eq. 2.19})$$

Where  $a$  is eigenvector corresponding to the maximum eigenvalue of a variance-covariance matrix of  $V_{xx}$  and evaluate  $a$  (eq. 2.18).

V. Dheepa et al. [77] and M. R. Lepoivre et al. [78] have invested the customer transaction behaviour with a technique of PCA and examined a fraud detection by SVM classifier.



The SVM model could make better classification between customers and frauds with the PCA features.

So far, I studied various methods of feature transformation for not only machine learning, but also deep learning and explored the possibility of taking the effective feature transformation methods in my framework. Through the whole studies in Section 2.4., I have learned that the feature engineering performed quite well for machine learning and deep learning and there is a potential to make more effective features by using both methods of feature aggregation and transformation. Next, I have explored methods on how to select effective features for improving ML/DL model's performance from all features in the dataset.

## **2.5. Feature Selection**

Feature selection is a method of removing the irrelevant, inconsistent, and redundant variables when developing a predictive model [86]. In some literatures, feature selection is considered as a feature engineering method. In most research, feature selection is referred to as feature reduction, which is also known as dimensionality reduction to avoid from overfitting [80] [82] [83].

In general, performances of machine learning algorithms are influenced by the training data. The important part of the success of building a good machine learning model is coming up with the good training data which contains adequate relevant features and not too many irrelevant ones [82]. There are two major reasons for the low performances of ML algorithms, which are overfitting and underfitting.

When a machine learning model gets trained with a huge data, the model learns the whole features including the noise and inaccurate data entities in the dataset. Then, the model will not be able to categorise the data precisely because the model covers all variances in many irrelevant features. It is called an overfitting model. Underfitting is the opposite of overfitting. It happens when the model is too simple to learn the fundamental structure of the training data whereas overfitting appears when the model is too complex relative to the values and noisiness of the training data. The performance of the underfitting model is poor because the model is too simple to classify the target.

Figure 2-29 and figure 2-30 show the problems of underfitting and overfitting and how a linear regression model with polynomial features can be used to approximate nonlinear functions. For instance, the model in Figure 2-30 is prone to underfit, which means the model's predictions incur inaccurate. On the other hand, the model in Figure 2-29 strongly overfits the training data, which indicates that the model performs much better on the training data than the true function.

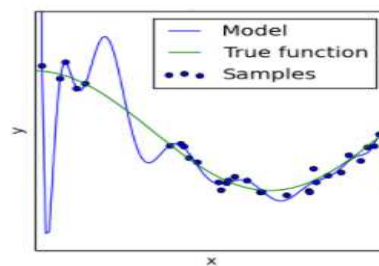


Figure 2-29. Overfitting with training data [81]

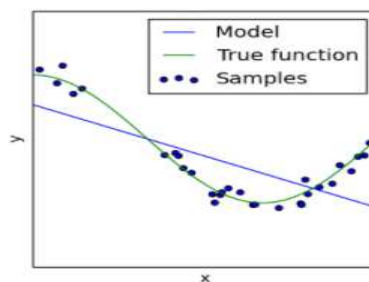


Figure 2-30. Underfitting with training data [81]

Complex models such as deep learning and neural networks can detect subtle patterns in the data, however, if the training dataset includes noisiness and irrelevant features, then a model is likely to detect patterns in the irrelevant and noise itself [80].

To avoid overfitting and underfitting, here are summarised possible solutions from the research [80] [82] [83]:

#### A) Preventing model overfitting

- Reduce the number of variables in training data when building a predictive model.
- Assemble more training data.
- Decrease the noise or irrelevance in training data by removing outliers or unrelated attributes.

#### B) Preventing model underfitting

- Select more effective features in training data.
- Build a predictive model with enough data.
- Select a more effective model with more parameters.

Figure 2-31 shows a good performance line model which covers majority of the samples in the data and maintains the balance between overfitting and underfitting.

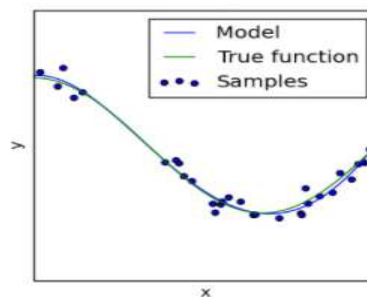


Figure 2-31. Appropriate fitting model [81]

In short, the good performance model needs to be fed with effective features in the training data which has the powerful valid features without the irrelevant features.

Using feature selection in training data will make a better predictive model.

R.C. Chen et al. [84] studied a feature selection process for credit card fraud detection. In order to select most influenced attributes related to a fraud detection system, they used the feature selection methods i.e., filter, wrapper and embedded methods to find an effective feature in unsupervised learning to discover a credit card fraud. The experiment was conducted based on six different unsupervised learning algorithms with the selected features and only raw features. The overall results were enhanced with the selected features rather than only using raw features.

C. E. Brodley et al. [85] proved the necessity for feature selection through their experiments with the development of an automated subset selection algorithm for fraud detection that employed the Expectation-Maximization clustering method that disperse separability and maximum likelihood.

Kajal Kamaljit Kaur [86] concluded that feature selection and balancing unbalanced label dataset should be carried out to enhance a credit card fraud detection for machine learning algorithms, for instance in the paper, they employed random forests, Naïve Bayes, Logistic regression, Multilayer proception NN, ANN. Through the whole results of experiments using the selected features is remarkably significant in achieving meaningful results.

All the Above studies have proven the effectiveness of using feature selection and improved accuracy of machine learning algorithms. However, in their studies, they only

focused on selecting effective features from a given dataset and do not use feature engineering from the aspect of creating effective candidate features for machine learning.

I have learned a lot from existing studies of using feature engineering or feature selection and understood their effectiveness for machine learning and deep learning specifically in financial fraud detection. Next, I explore existing studies of a new feature engineering framework that is used in not only fraud detection cases but also any classification cases to learn the general impact of using the feature engineering framework.

## **2.6. Feature Engineering Tools and Framework**

The concepts and individual techniques of feature engineering were reviewed and studied in the variety of areas. Through the reviews, the effectiveness of using feature aggregation or feature transformation methods were proven. Below, some studies of using feature engineering tools and framework are highlighted.

K. Veeramachaneni [20] et al. developed Deep Feature Synthesis (DFS) tool that generates lots of features by aggregating features in a relational database structure. The DFS acts feature engineering for multi-table and transactional datasets generally found in datasets. Many features are created by using the fundamental statistic: average, sum, maximum, minimum and standard deviation. The input to DFS is a set of related entities and the tables associated with them. Figure 2-32 illustrates the concept of deep feature synthesis.

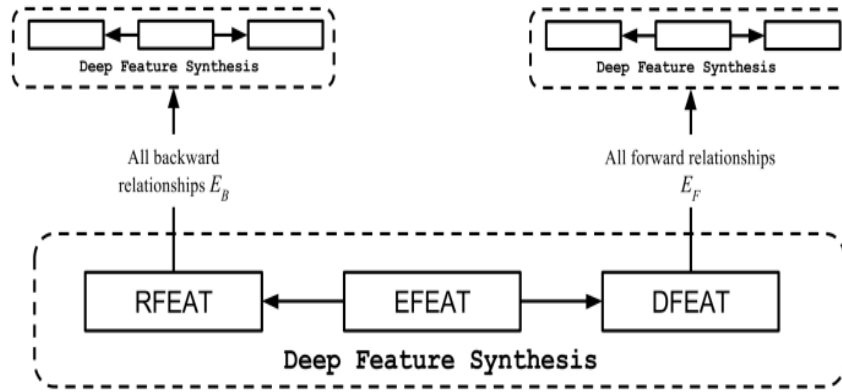


Figure 2-32. Demonstration of the concept Deep Feature Synthesis [20]

Both *REFEAT* and *DEFEAT* features can be synthesized independently while *EFEAT* features depend on both *REFEAT* and *DEFEAT* features.

The first features set is computed by considering the features and their values in the table that is called “entity features”. *EFEAT* (Entity features) derives features by calculating a value for each entry  $x_{i,j}$ . These features should be based on the calculation function applied elementwise to the array  $x_{:,j}$ , and they also include applying a function to the entire set of values for the  $j^{th}$  feature,  $x_{:,j}$ , and  $x_{i,j}$ , given by:

$$x'_{i,j} = \text{efeat}(x_{:,j}, i) \quad (\text{eq. 2.20})$$

There are other two entities of  $E^l(\text{Forward})$  and  $E^k(\text{backward})$  which relate to each other. A forward relationship is between an instance  $m$  of entity  $E^l$ , and a single instance of another entity  $i$  in  $E^k$ . This is analysed the forward relationship between  $i$  has an explicit dependence on  $m$ . A backward relationship is from an instance  $i$  in  $E^k$  to all the instances  $m = \{1 \dots M\}$  in  $E^l$  that have forward relationship to  $k$ .

Direct Features (*dfeat*) is applied over the forward relationships. Features in a related entity  $i \in E^k$  are transferred as features for the  $m \in E^l$ . Relational Features(*rfeat*) is applied over the backward relationships and is derived for an instance of  $i$  of entity  $E^k$  by

applying a mathematical function to  $x_{:,j|e^k=i}^1$ , which is a collection of values for feature  $j$  in related entity  $E^l$ , gathered by extracting every values for feature  $j$  in entity  $E^l$  where the identifier of  $E^k$  is  $e^k = i$ . This transformation is provided by:

$$x_{i,j'}^k = rfeat \left( x_{:,j|e^k=i}^1 \right) \quad (\text{eq. 2.21})$$

Figure 2-33 shows the algorithm how to generate features for target entity in the DFS.  $F^i$  is presented to make features for the  $i^{th}$  entity. The organisation of repeated calls and calculation of each feature type is in accordance with the restrictions explained in the algorithm.

---

**Algorithm 1** Generating features for target entity

---

```

1: function MAKE_FEATURES( $E^i, E^{1:M}, E_V$ )
2:    $E_V = E_V \cup E^i$ 
3:    $E_B = \text{BACKWARD}(E^i, E^{1..M})$ 
4:    $E_F = \text{FORWARD}(E^i, E^{1..M})$ 
5:   for  $E^j \in E_B$  do
6:     MAKE_FEATURES( $E^j, E^{1..M}, E_V$ )
7:      $F^j = F^j \cup \text{RFEAT}(E^i, E^j)$ 
8:   for  $E^j \in E_F$  do
9:     if  $E^j \in E_V$  then
10:      CONTINUE
11:    MAKE_FEATURES( $E^j, E^{1..M}, E_V$ )
12:     $F^i = F^i \cup \text{DFEAT}(E^i, E^j)$ 
13:   $F^i = F^i \cup \text{EFEAT}(E^i)$ 

```

---

Figure 2-33. The algorithm of Deep Feature Synthesis [20]

The *REFEAT*, *DFEAT*, and *EFEAT* functions in Figure 2-33 are responsible for synthesizing their respective feature types based on the given input.

Figure 2-34 demonstrates an example of a feature that would be created by the DFS tool and displays how features are computed by traversing relationships between entities.

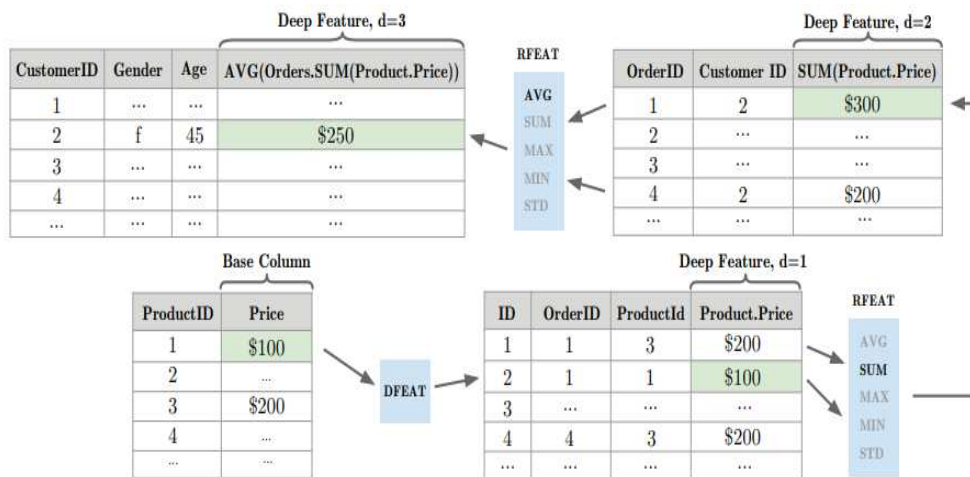


Figure 2-34. Sample case of using DFS tool [20]

The Deep Feature Synthesis is a feature engineering tool useful for generating many features productively and effectively for machine learning algorithms. However, this tool is not built for a fraud detection case but for a general use.

R. Wedge et al. [79] suggested using the DFS tool that creates new attributes for machine learning models of credit card fraud detection using the relational structure of the dataset. In the processes of DFS, both feature aggregation and transformation methods are used to create new features using attributes of the related transactions. For instance, they applied the Hour in transaction time to determine when a transaction has occurred during the day and use statistical methods i.e., average, mean, sum and standard deviation to express the user behaviour on the transaction time base. Timestamps in transactions are significant processes in DFS to compute features of every month and within 24 hours as shown in Figure 2-35.



Features aggregating information from all the past transactions	
Expression	Description
cards.MEAN(transactions.amount)	Mean of transaction amount
cards.STD(transactions.amount)	Standard deviation of the transaction amount
cards.AVG.TIME_BETWEEN(transactions.date)	Average time between subsequent transactions
cards.NUM_UNIQUE(transactions.DAY(date))	Number of unique days
cards.NUM_UNIQUE(transactions.tradeid)	Number of unique merchants
cards.NUM_UNIQUE(transactions.mcc)	Number of unique merchant categories
cards.NUM_UNIQUE(transactions.acquirerid)	Number of unique acquirers
cards.NUM_UNIQUE(transactions.country)	Number of unique countries
cards.NUM_UNIQUE(transactions.currency)	Number of unique currencies

Figure 2-35. Features creation using the DFS tool [79]

In Figure 2-38, each feature aggregates data belonging to previous transactions from the credit card. The left column displays how the feature is calculated. The right column explains what the feature means. At the end, they generated 237 features (over 100 behavioural pattern features) for each transaction and reduced the false positive rate by 54%. However, in their study, they used all 237 generated features without feature selection.

Another feature engineering framework is Automatic Feature Generation and Selection which is also called ExploreKit. G. Katz et al. [19] developed the framework that generates a large set of candidate features, with the aim of maximizing performance of ML models according to user-selected criteria. They employed the approach of machine learning-based feature selection which predicts and selects the useful new created features from a large set of candidate features as seen in Figure 2-36.

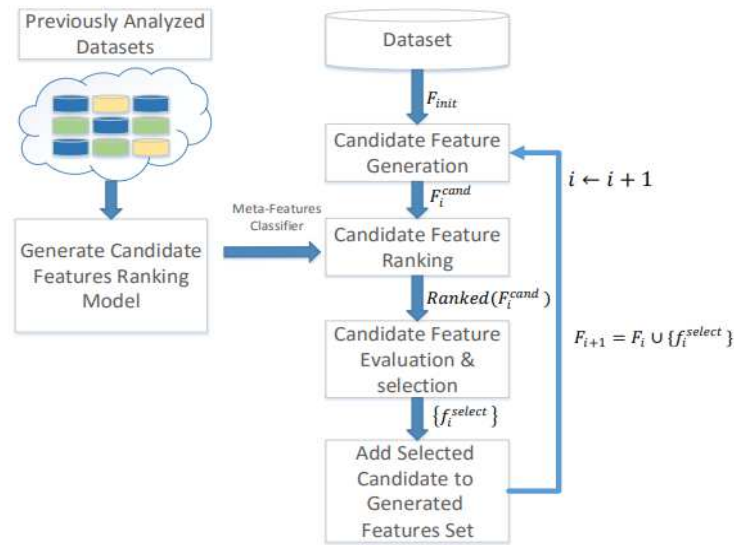


Figure 2-36. ExploreKit system architecture [19]

The ExploreKit framework includes two main parts, generation of candidate features and ranking candidate features. The aim of generating candidate features is to generate a large set of candidate features  $F_i^{cand}$  using the present features set  $F_i$ . There are three types of operators to create candidate features set  $F_i^{cand}$  for iteration  $i$ , which are unary, binary and higher order. Unary operators implement on a single feature and uses discretizers which are used to convert continuous into discrete ones, and normalizers which are used to fit the scale of continuous features to specific distributions. In this paper, the EqualRanged discretization for numeric features (partition on the range of values of the feature into X equal segments) is implemented.

Next, binary operators are applied on a pair of features and consist of the four basic arithmetic operations such as  $+$ ,  $-$ ,  $\times$ ,  $\div$ . Finally, Higher order operations utilise multiple features for the generation of a new one and implement five operators such as Max, Min, Average, Standard Deviation, and Count. Another main part in the ExploreKit is ranking candidate features. In this part, feature importance is used to rank the large number of candidate features  $F_i^{cand}$ . In their experiment, they demonstrated the effectiveness of

using the ExploreKit on multiple datasets by leading an extensive evaluation with 3 different ML algorithms for classification. They managed to reduce 20% classification-error overall when using the framework. However, they only tested on the general datasets that are a well-balanced target numbers and are not related to financial fraud. The ExploreKit has a process of feature generation, but it only uses arithmetic operation for increasing the number of new features without customer's behaviour analysis.

## **2.7. Summary & Conclusion**

The research carried out as part of the literature review outlined a huge number of the related studies both part of feature engineering methods and fraud detection algorithms such as supervised, unsupervised, and deep learning in the financial area. First, the research of machine learning algorithms in financial institutions was conducted to develop an understanding of a fraud detection problem and to learn how they treat input data in each algorithm. Basically, machine learning algorithms learnt different tendencies between normal transactions and fraudulent transactions from given features in a dataset by using various calculation methods. Most of the studies were carried out in the field of credit card fraud detection and they have focused on increasing the accuracy of fraud detection by using advanced machine learning techniques instead of elaborating input features. In financial fraud detection, most studies of using feature engineering methods were about feature aggregation based on customer's behaviour during a transaction. Deep learning uses a structure of multiple layers where each input feature combines each other and unifies as one feature in the next layer. To avoid the curse of dimensionality, feature engineering method in deep learning is used for the purpose of dimensionality reduction rather than the purpose of clarifying the latent patterns of input

data. Most studies using deep learning in financial fraud detection have not applied feature aggregation methods on the input data with an aim of new feature creation. In Chapter 2, I introduced three types of the most representative deep learning algorithms: autoencoder neural networks (AEs), convolutional neural networks (CNNs), recurrent neural networks (RNNs). The fundamental concept of their structures was very similar in terms of a use of hidden layers and reconstruction of the input data by computing weights between features. They also implement feature engineering when calculating each weight for generating new features in hidden layers. Feature selection is a process to select a subset of features from a dataset and is considered as a feature engineering method. In most studies, feature selection is used for avoiding overfitting and low accuracy. When a machine learning model gets trained with a large amount including the noise and irrelevant data, this will cause worse performance of machine learning models.

Throughout the literature reviews in Chapter 2, they proved that applying feature engineering and feature selection methods to an original dataset for training machine learning algorithms improves the prediction accuracy of the models. They provided the evidence of effectiveness of using feature aggregation, feature transformation, feature selection individually. However, they have not used the whole methods simultaneously in one study. Furthermore, they seldom used the feature engineering methods for deep learning. Hence, this is the way I need to build a new framework that consists of both methods of feature creation and feature selection in a series of feature engineering processes and the framework also can provide the most effective features set for not only machine learning but deep learning.

## 2.8. Literature Synthesis

This section describes the applicable literature that has been consolidated in creating the new feature engineering framework as shown in Table 2-3.

Literature	How it is used	Section
Machine Learning Algorithms	There are many kind of machine learning algorithms are introduced to learn how input features are dealt in each algorithm. Our main study is feature engineering which generates new features for improving the ML model's performance. Therefore, it is necessary to understand the structure of data processes in ML algorithms.	Data Preparation Feature Selection
Support Vector Machine	The SVM model is trained by using both only raw dataset as a baseline model and the created new feature set for evaluating the effectiveness of using the new feature set.	Evaluation Model SVM
Random Forests	The RF model is built with all features set for measuring feature importance.	Feature Selection Feature Importance
Artificial Neural Networks	ANNs is used for understanding as basic concept of deep learning and how to work with input features	ANNs Deep Learning
Autoencoder Neural Networks	Autoencoder is selected as the representative of a deep learning model for financial fraud detection and the autoencoder model is built with both only raw dataset and the created new features set for evaluating the effectiveness of using the new feature set.	Evaluation Model AEs
Conventional Neural Networks	CNN is covered for learning the possibility of using deep learning for financial fraud detection.	Deep Learning CNNs
Recurrent Neural Networks	RNN is covered for learning the possibility of using deep learning for financial fraud detection.	Deep Learning RNNs
Clustering	Clustering is used as part of the feature engineering technique and as part of the standardisation technique.	Feature Engineering K-mean PCA

Isolation Forest	IF algorithm is selected as the representative of unsupervised learning and it is built with both only raw dataset as a baseline model and the created new features set for evaluating the effectiveness of using the new feature set.	Evaluation Model IF model Feature selection
Local Outlier Factor	LOF algorithm is selected as the representative of unsupervised learning, and it is built with both only raw dataset as a baseline model and the created new features set for evaluating the effectiveness of using the new feature set.	Evaluation Model LOF model Feature Selection
Feature Aggregations	Feature aggregation is used as part of feature creation method.	Feature Creation
Feature Transformations	Feature transformation is used as part of feature creation method.	Feature Creation
Feature Engineering Framework	Feature engineering framework is used to learn the latest technique and other techniques of feature engineering	Feature Engineering Framework
Feature Selection	Feature selection is used for understanding why other studies have applied the feature selection techniques among the dataset and how they worked.	Feature Importance

*Table 2-3. The summary of literatures and how they are used in my research work*

# 3. Feature Engineering Framework for Financial Fraud Detection Models

## 3.1. Overview

As discussed in Chapter 1, both financial institutes and academic studies have tackled topics pertinent to the fraud issues using advanced machine learning methods to detect fraudulent actions more certainly. However, the total losses through online banking in the United Kingdom have still increased because fraudulent techniques have progressed and used advanced technology. It is difficult to expose fraudulent behaviour patterns by only using raw data extracted from the linked tables to transactions. In Chapter 2, the current studies of various approaches using machine learning and deep learning algorithms for financial fraud detection are highlighted and studied in terms of the process of how to deal with the input data. Then, the current studies of feature engineering methods for classification are explored.

Through the overall reviews in Chapter 2, it was convincing that using features created by feature engineering methods is very effective for improving performance of fraud detection models and it was evident that there are different weaknesses that need addressing. The intent of my research is to encourage the use of an effective features set generated by the new feature engineering framework that contains both processes of feature creation and feature selection.

Figure 3-1 displays a feature engineering framework that demonstrates the series of processes in the feature creation workflow and the feature selection workflow.

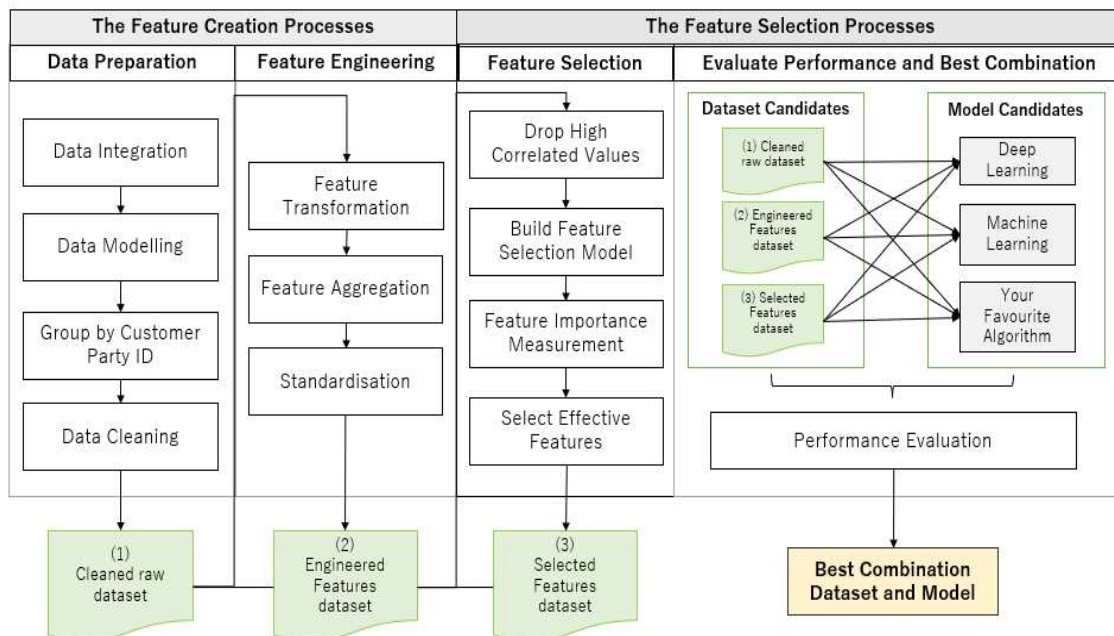


Figure 3-1. Conceptual Feature Engineering Framework

The framework shown in Figure 3-1 consists of the following processes:

- The aim of the feature creation processes is to create new effective features for a predictive model. To achieve it, first, the collected raw data needs to be organized and arranged as part of a data preparation workflow. Then, feature engineering techniques are applied to clean data. Eventually, two datasets will be prepared through the feature creation processes. First one is a simple raw dataset that is cleaned and arranged for building a baseline model. Another one is an engineered features set that includes both raw attributes and new features created by the feature engineering techniques: feature aggregation and feature transformation. Further elaboration can be found in Section 3.2.
- The aim of the feature selection processes is to exclude irrelevant features and high correlated values from the dataset based on two types of feature performance indicators: correlation coefficient and feature importance. The selected features



dataset is used for building a predictive model as the most effective performance model compared to the other datasets: a raw dataset and the dataset with all features. This section will also cover processes of the performance evaluation of each model.

## 3.2. Feature Creation Processes

Under the feature creation processes, there are two main parts to create new features, namely data preparation and feature creation.

### 3.2.1. Data Preparation

#### I. Data Integration

Data quality and accessibility are the most important part for machine learning and will make an impact on model accuracy of the model. With the online payment system advances, transaction data in the digital world becomes much better to access. Also, the related data with transaction can be collected from the various sources such as different payment channels, web service, portable devices, flat files, and payment digital applications [57] as shown in Figure 3-2.

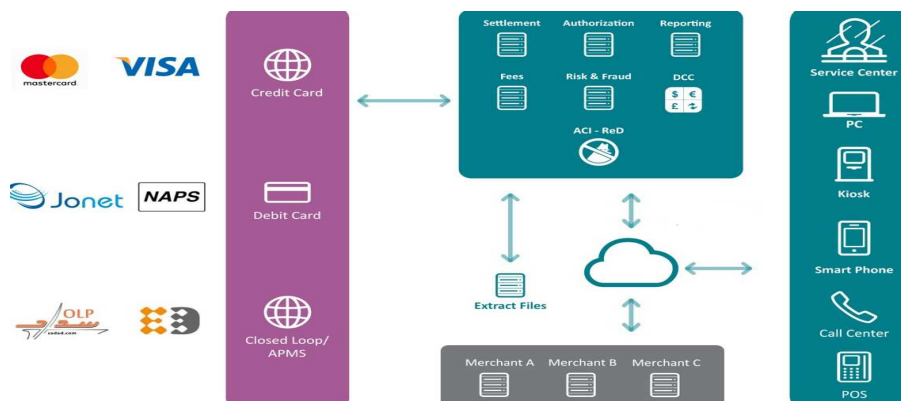


Figure 3-2. Example of different sources [57]

As drawn in Figure 3-2, transactions are linked to multiple data sources such as the information of credit/debit cards, commercial institutions or banks where financial services to customers, internet/online services, portable/remote accessing devices, locations, and online/offline shops are provided and need to be collected in one place to demonstrate customer's transaction behaviour.

Data integration is the process of putting data together from different sources into a single unified place and assists both online and offline data collection. As described in Figure 3-2, various types of customer's queries for transactions or transfer from the bank account are integrated into the banking system. The diagram in Figure 3-3 shows the logical flow of data in the system, processes and data sources related to the transaction.

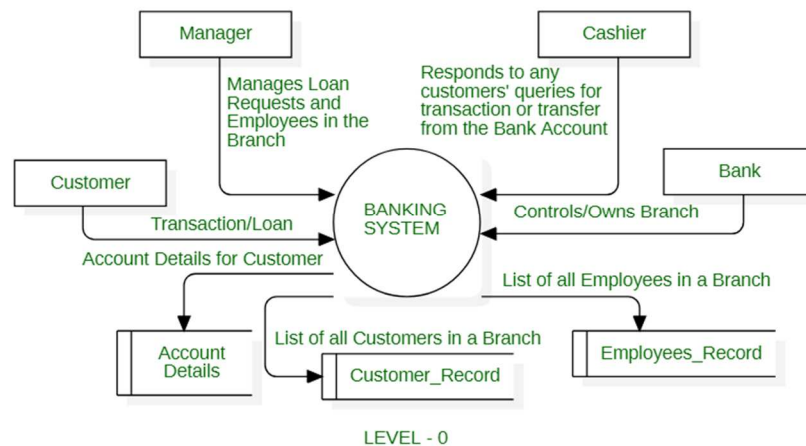


Figure 3-3. Image of data integration into the banking system [105]

In many studies related to financial fraud detection, although their datasets for their experiments were provided from private financial institutions or downloaded from Opensource and they were already integrated into one table from different sources, data for analysing fraudulent transaction should be collected based on a plan which attributes can be extracted and how to integrate them into a one table.

## II. Data Modelling

Data modelling is the process of designing a clarified diagram of data tables from different sources using data elements and flows and providing a scheme for merging the discrete attributes into a new data format to be stored in a database. Figure 3-4 below shows a database schema of the logical data model for online banking system.

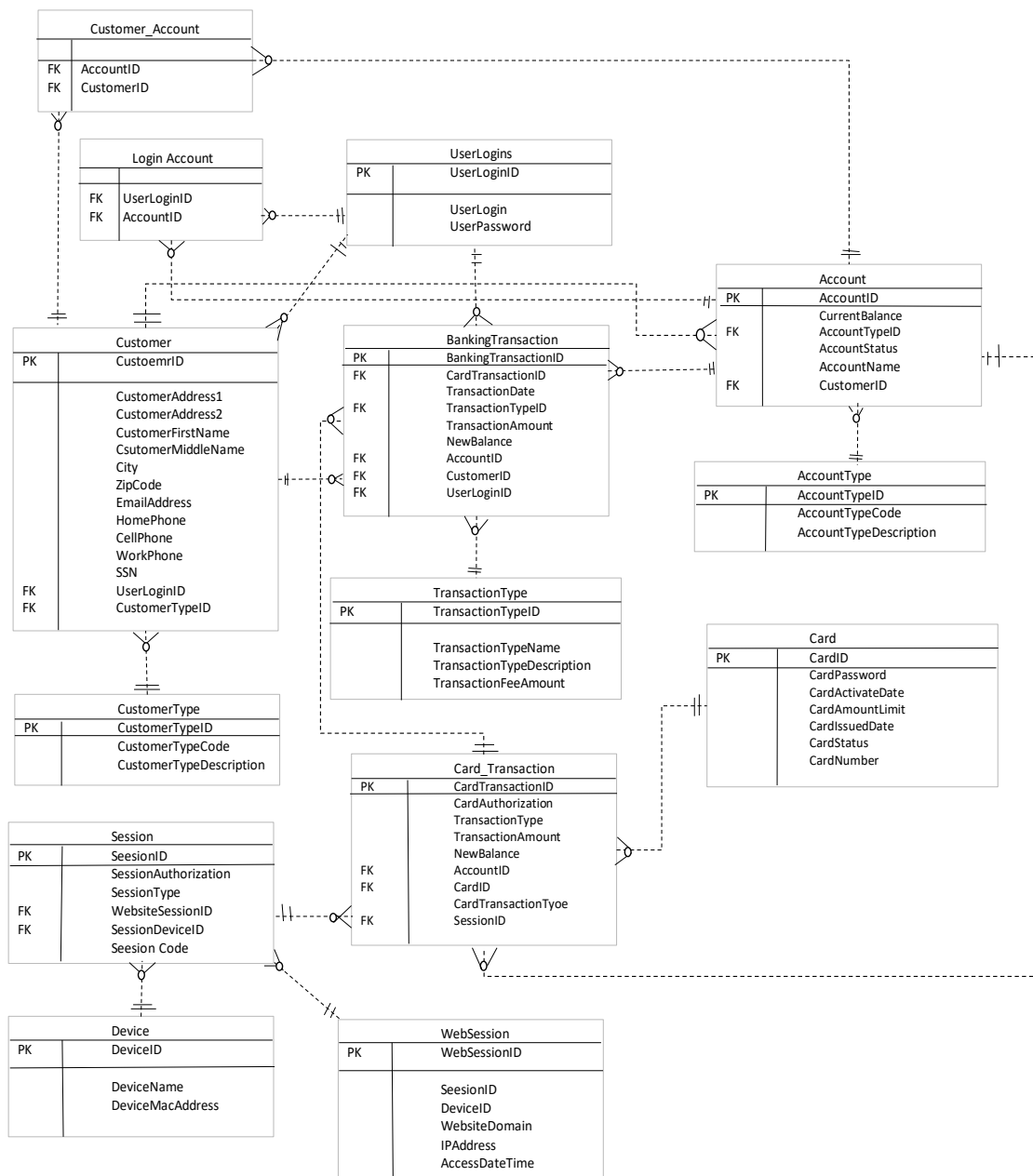


Figure 3-4. Logical Data modelling of Online Banking System

The reason why it is important to understand the data modelling in the research is because attributes used for building a detection model will be extracted from the banking database tables based on the domain knowledge of a data engineer who determines which attributes are useful. In this research, although a bunch of online banking data provided by a European private bank was already extracted from the system, all attributes related to user's behaviour on transaction should be considered as to whether they need to be collected for making a good fraud detection model.

### **III. Group by Customer Party ID**

Raw data collected from various sources will be messed and have many missing values. Thus, the raw data needs to be cleaned. Before data cleaning, the data needs to be grouped by customer base so that it is necessary to analyse and fill in missing values in each attribute by individual customer's tendency. While a customer uses an online banking system, each customer may have his/her own transaction behaviour or payment patterns based on regular accessing time, common device and IP address, a regular payment, and amounts. Therefore, if missing values are filled with all averages of all customer's variables in the attribute, the data will affect the performance of the detection model. In order to deal with the missing values more appropriately in this research, the dataset will be arranged by grouping customer ID base before data cleaning. More detailed techniques of handling missing values are studied and described in the next paragraph.

### **IV. Data Cleaning**

The processes of data cleaning have mainly two operations: converting character string to numeric data type and coping with missing, noise or wrong values.

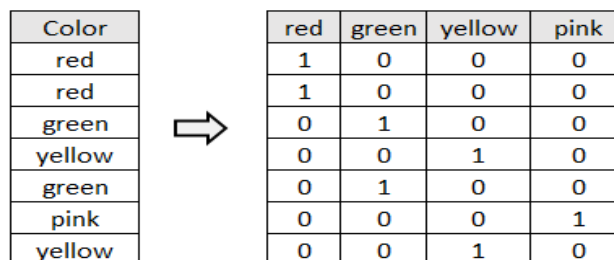
### **(A) Convert categorical feature's datatype:**

A categorical variable is used to represent categories or labels as the name implies. For example, a categorical variable would describe major cities in the world, the industry of a company, color types in a product, or gender types in the personal information. The number of categorical values can be represented numerically. However, the values of a categorical variable cannot be ordered regarding one another. For instance, green is neither greater than nor less than red as a color type. These types of categorical values are called nonordinal. Other categorical variables are interval variables which indicate between two things that define the spectrums of values for measurement points, for instance, income attribute will be expressed as the ranges of 0-1,000, 1,000-2,000 and more, which can be represented numerically. There are also large categorical variables such as IP addresses, transaction IDs, or customer IDs, which are categorical values with over hundreds values of unique users. Although IP addresses and customer IDs are numeric, their size is normally not relevant to the task at hand. For instance, the IP address should be relevant while doing fraud detection on each transaction. Consequently, some IP addresses may create more fraudulent transactions than others. However, a subnet of 147.199.x.x is not constitutionally more fraud than 147.200.x.x. In other words, the subnet number does not matter. There are some techniques to convert categorical variables to numerical variables appropriately [56] [55] [54].

There are some categories, which cannot be usually numeric variables. For instance, hair color can be "black", "blond", "brown", etc. Therefore, an encoding method is required to turn these nonnumeric categories into numbers. There are some encoding methods to convert string values to numeric values. Thus, I investigate these methods and study how they work, then determine which method is appropriate to a purpose of my research.

### i. One-Hot Encoding

A progressing method is to utilise a group of bits. Each bit describes a feasible category. If the variable cannot be part of multiple categories at once, then only one bit in the group can be “1”. This is called one-hot encoding (OHE). Each of the bits become a feature. Therefore, a categorical variable with  $N$  possible categories is encoded as a feature vector of length  $N$ . Figure 3-5 shows an example.



Color	red	green	yellow	pink
red	1	0	0	0
red	1	0	0	0
green	0	1	0	0
yellow	0	0	1	0
green	0	1	0	0
pink	0	0	0	1
yellow	0	0	1	0

Figure 3-5. Example of one-hot encoding method [56]

OHE is quite easy to understand, however it applies one or more bit than is rigidly necessary. If I notice that  $N-1$  of the bits are 0, then the last bit must be 1 because the variable should take on one of the  $N$  values. This restriction is described so that the sum of all bits should be equal to 1.

$$e_1 + e_2 + \dots + e_N = 1 \quad (\text{eq. 3.1})$$

Therefore, I can see a linear dependency on this equation. Linear dependent features are a little disrupting because they imply that the linear models will not be unique. Consequently, it becomes difficult to understand the effect of a feature on the prediction.

## ii. Dummy Coding

The issue of using OHE method is that it gives for  $N$  degrees of freedom, while the variable itself calls for only  $N-1$ . Dummy coding eliminates the additional degree of freedom by utilising only  $N-1$  features in the description as shown in Figure 3-6.

Color	red	green	yellow	pink
red	1	0	0	0
red	1	0	0	0
green	0	1	0	0
yellow	0	0	0	0
green	0	1	0	0
pink	0	0	0	1
yellow	0	0	0	0

Figure 3-6. Sample of dummy coding [56]

One feature (Yellow in Figure 3-6) is described by the vector of all zeros. The result of modelling with dummy coding is more explicable than with OHE. This is clear to see in a simple linear regression issue. For instance, some data about the information of land prices in three cities: London, Liverpool, and Manchester are shown in Table 3-1.

Index	City	Land price (£)
0	London	340,000
1	London	500,000
2	London	280,000
3	Liverpool	800,000
4	Liverpool	100,000
5	Liverpool	160,000
6	Manchester	220,000
7	Manchester	700,000
8	Manchester	145,000

Table 3-1. Information of land prices in three cities [19]

I trained a linear regressor to predict land price based solely on the identity of the city.

The linear regression model can be described as:

$$y = w_1x_1 + \dots + w_nx_n \quad (\text{eq. 3.2})$$

It is customary to fit the intercept in order that  $y$  can be a nonzero value while the  $x'_s$  are zeros and the  $w'_s$  are weights.

$$y = w_1x_1 + \dots + w_nx_n + b \quad (\text{eq. 3.3})$$

Table 3-2 describes the implementation of OHE on the categorical variables.

	Land price	City_Liverpool	City_London	City_Manchester
0	340000	0	1	0
1	500000	0	1	0
2	280000	0	1	0
3	800000	1	0	0
4	100000	1	0	0
5	160000	1	0	0
6	220000	0	0	1
7	700000	0	0	1
8	145000	0	0	1

Table 3-2. Categorical variables in City were converted to one-hot encoding

	Land price	City_London	City_Manchester
0	340000	1	0
1	500000	1	0
2	280000	1	0
3	800000	0	0
4	100000	0	0
5	160000	0	0
6	220000	0	1
7	700000	0	1
8	145000	0	1

Table 3-3. Categorical variables in City were converted to dummy coding

Table 3-3 describes the implementation of dummy coding on the categorical variables. Then, I fit a linear regression model by using a package of a sklearn python library on each data. The results of weights and intercepts in each model are shown in Table 3-4. With one-hot encoding, the intercept phrase illustrates the overall mean of the target variable, Land price, and each of the linear coefficients illustrates how much that land's average price differs from the overall mean. With dummy coding, the bias coefficient illustrates the mean value of the target variable  $y$  for the reference category, whose instance is City Liverpool. The coefficient for the  $i^{th}$  feature is



equivalent to the difference between the mean target value for the  $i^{th}$  category and the mean of the reference category. Table 3-4 shows how these encoding methods generate quite different coefficients for linear regression models.

	<b>w1</b>	<b>w2</b>	<b>w3</b>	<b>B</b>
<b>One-Hot Encoding</b>	-7222.22	12777.77	-5555.55	360555.55
<b>Dummy Coding</b>	0	20000	1666.66	353333.33

Table 3-4. Learned coefficients by linear regression

### iii. Effect Coding

Another method of encoding categorical variable is effect coding. Effect coding is quite similar to dummy coding, but it uses “-1” for representing the reference category instead of zeros. Table 3-5 describes the implementation of effect coding on the categorical variables.

	<b>Land price</b>	<b>City_London</b>	<b>City_Manchester</b>
<b>0</b>	340000	1	0
<b>1</b>	500000	1	0
<b>2</b>	280000	1	0
<b>3</b>	800000	-1	-1
<b>4</b>	100000	-1	-1
<b>5</b>	160000	-1	-1
<b>6</b>	220000	0	1
<b>7</b>	700000	0	1
<b>8</b>	145000	0	1

Table 3-5. Categorical variables in City were converted to effect coding

The advantage of using effect coding is to give a simpler interpretation of results in linear regression models. The intercept phrase illustrates the overall mean of the target variable, and each coefficient expresses how much the means of each category differ from the overall mean. OHE method discovered the same intercept and coefficients, however in this case, there are linear coefficients for each city. In the case of effect coding, no single feature indicates the reference category. Thus, the effect of the

reference category requires to be separately calculated as the negative sum of the coefficients of any other categories.

I introduced the three major methods of encoding for categorical variables: One-hot encoding, dummy coding, effect coding. They are very similar to one another and have pros and cons respectively. Dummy coding and effect coding provide rise to unique and interpretable models whereas one-hot encoding allows for multiple valid models for the same problem. The advantage of OHE is that each feature apparently corresponds to a category. Furthermore, missing values will be encoded as all zero values, and the output would be the global mean of the target variable. The disadvantage of dummy coding is that it cannot simply deal with missing data because all zero values are already mapped to the reference category. In contrast, effect coding uses a different code for the reference category. However, handling the vector of many -1s will take a highly computation cost and need large storage. Therefore, effect coding is seldom selected to use in the most ML studies due to the fact that it is expensive.

However, all three methods are not suitable for categorical values when the number of categories becomes large such as the user ID, IP addresses, software version and so on. The challenge is to find a better method for encoding categories that is efficient but is not costly.

#### **iv. Label Encoding**

Label encoding is quite simple approach and converts each categorical variable to a number. It encodes labels with a value between 0 and  $N$  where  $N$  is the number of discrete labels. Consider below the sample in Table 3-6:

ID	City
1	London
2	Liverpool
3	Manchester
4	Liverpool
5	Bristol
6	Canterbury
7	Cambridge
8	London
9	Bath
10	Manchester

*Table 3-6. Sample of categorical variables*

If I use this data to train a machine learning algorithm, the city attribute needs to be encoded to the numeric variables, and in this case, I use label encoding. A package of label encoding method is available from Scikit-learn and it is easy to use. After applying a label encoding method, the result becomes as seen in Table 3-7 below.

ID	City
1	1
2	2
3	3
4	2
5	4
6	5
7	6
8	1
9	7
10	3

*Table 3-7. Converted into numerical variables by using label encoding*

Label encoding is a simple and easy way to convert character values to numerical values. Such numerical values are arbitrary, however. In some cases, it is more efficient to assign specific numbers. For instance, each city or town has an official UK area code e.g., 020 for London, 0117 for Bristol, etc. [97]. If the area codes are assigned to the city names as shown in in Table 3-6, such data can be connected to other data that use the same area codes, such as demographics, income, etc.

Country names can be also represented by country codes e.g., US 1, UK 44, Japan 81, etc and they can keep their own meaning as it has even though they are converted to numerical values. Another example is city feature values such as East London, Central London, Edinburgh, Enfield, etc can convert into individual postcodes that are commonly used as a representation of the area information and can provide more additional information by connecting each postcode such as the information of local communities, real property, and the public peace, where generally can be grouped by each postcode. The connection image is described in Figure 3-7.

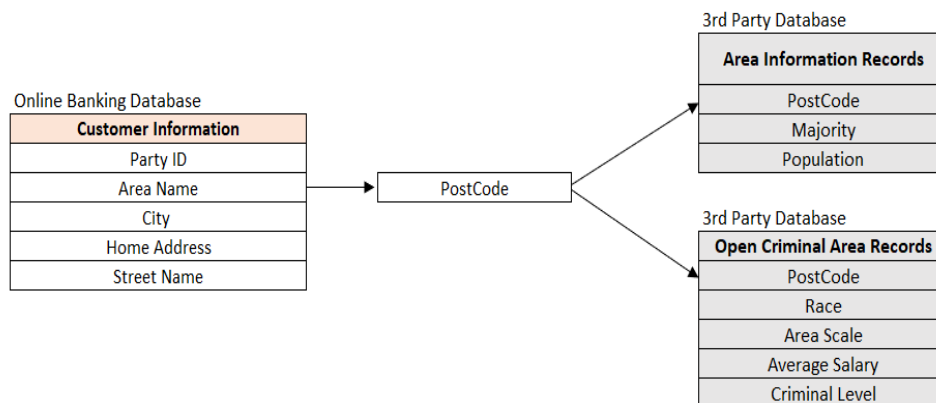


Figure 3-7. Specific number can connect to other information tables

There will be another chance to obtain more rich features related to customer's information. Thus, If I assign random numbers to place name features with encoding methods, then these features will lose original meaning and other connection.

**(B) Types of missing data analysis:**

One of the hard tasks while dealing with raw data is missing values. Usually, raw data is not organised and has a lot of missing values because of human errors, privacy concerns, disruptions and so on. Missing values are shown as blank in a dataset displayed in Figure 3-8. There are roughly two types of missing values. The first type of missing data is completely independent of other values. There is no relationship between the missing

values and any other values in the dataset, which means that there is no pattern of being missing values. In this case, the data became missing because of human error or some system failure while logging the data. This type of missing values can be dealt with unbiased analysis. The next type is that missing values are only subsamples of the data. There is some relationship between the missing values and other values or data, and there is some pattern of being missing values. For instance, there are some categories in the Approach Methods attribute such as Website, mobile apps, phone call, and visit. If there are missing values under the IP Address attribute, the missing values may indicate that they are missing due to phone call or visit in the Approach Methods attribute. Thus, in this case, the missing value of the IP address should be substituted by the specific number which differs from other IP addresses.

EVENT	Event Type	TRANSACTION TIME	AUTHENTICD	CLIENTSCREENID	INTERNET_ACCESS	INTESESSIONID	IPADDRESS	DVD_TZDT	USERAG	VD_USER	LATENCY
www.NotifySe	MIDA_EVENT	Apr 5, 2016 8:19:00 PM	500	1366x768	Apr 5, 2016 8:19:00 PM	ab52f6311e5b158b7119c55	86.172.214.248	0	WOW64;	9.23E+08	92
www.NotifySe	MIDA_EVENT	Apr 6, 2016 9:56:52 AM	500	1366x768	Apr 6, 2016 9:56:52 AM	8572fbd511e5a20a45cc7cfc	86.172.214.248	0	WOW64;	9.23E+08	128
www.NotifySe	MIDA_EVENT	Apr 6, 2016 10:02:25 AM	500	1366x768	Apr 6, 2016 10:02:25 AM	318cfbd611e5a5ac03c0cbd2	86.172.214.248	0	WOW64;	9.23E+08	110
www.NotifySe	MIDA_EVENT	May 11, 2016 10:48:26 AM	500	1366x768	May 11, 2016 10:48:26 AM	7006175d11e6b320435dc746	86.158.88.153	0	WOW64;	9.23E+08	136
www.NotifySe	MIDA_EVENT	May 11, 2016 10:54:35 AM	500	1366x768	May 11, 2016 10:54:35 AM	49f4175e11e6b801539fb17a	86.158.88.153	0	WOW64;	9.23E+08	130
www.NotifySe	MIDA_EVENT	May 17, 2016 9:02:12 AM	500	1366x768	May 17, 2016 9:02:12 AM	6bb81c0511e69d23e5826579	86.162.151.230	0	WOW64;	9.23E+08	96
www.NotifySe	MIDA_EVENT	May 17, 2016 9:39:07 AM	N								68
www.NotifySe	MIDA_EVENT	May 18, 2016 9:29:00 AM	500	1366x768	May 18, 2016 9:29:00 AM	e121cd211e69d23e5826579	86.162.151.230	0	WOW64;	9.23E+08	96
www.NotifySe	MIDA_EVENT	May 18, 2016 9:37:33 AM	500	1366x768	May 18, 2016 9:37:33 AM	4e981cd311e68d21e3704556	86.162.151.230	0	WOW64;	9.23E+08	70
www.NotifySe	MIDA_EVENT	May 28, 2016 9:02:37 PM	500	1366x768	May 28, 2016 9:02:37 PM	9b94250f11e69e1c536934f0	86.180.95.246	0	WOW64;	9.23E+08	107
www.NotifySe	MIDA_EVENT	May 29, 2016 8:23:03 PM	500	1366x768	May 29, 2016 8:23:03 PM	525c25d211e6836c414b8203	86.169.161.158	0	WOW64;	9.23E+08	122
www.NotifySe	MIDA_EVENT	Jun 2, 2016 5:32:44 PM	500	1366x768	Jun 2, 2016 5:32:44 PM	9f7ac28df11e69b412ff520a7	86.154.224.45	0	WOW64;	9.23E+08	113
www.NotifySe	MIDA_EVENT	Jun 3, 2016 4:29:43 PM	500	1366x768	Jun 3, 2016 4:29:43 PM	e316299f11e6b215275cb47	86.154.224.45	0	WOW64;	9.23E+08	132
www.NotifySe	MIDA_EVENT	Jun 3, 2016 4:39:22 PM	500	N	Jun 3, 2016 4:39:22 PM	4d9629a111e6bfc2b3d993e	86.154.224.45	0	WOW64;	9.23E+08	137
www.NotifySe	MIDA_EVENT	Jun 3, 2016 4:56:21 PM	500	1366x768	Jun 3, 2016 4:56:21 PM	eca829a311e6836c414b8203	86.154.224.45	0	OW64; rv4;	9.23E+08	120
www.NotifySe	MIDA_EVENT	Jun 3, 2016 5:22:54 PM	N								90
www.NotifySe	MIDA_EVENT	Jun 3, 2016 5:59:53 PM	CVP								85
www.NotifySe	MIDA_EVENT	Jun 3, 2016 6:16:48 PM	CVP								87
www.NotifySe	MIDA_EVENT	Apr 4, 2016 9:41:57 AM	900	640x1136	Apr 4, 2016 9:41:57 AM	3a1afa4111e5a12ce2da89d2	2.29.6.32	0	S(X) Apple	phone2502	183

Figure 3-8. Blanks indicates missing values in dataset

In many studies of financial fraud detection, they have not dealt with missing values correctly because they have utilised the prepared dataset provided by a cooperative financial institution or an open-data which was already cleaned and processed. However, a real-life data will include missing values that will affect the performance of the ML models if they are not dealt with appropriately. There are mainly three options to deal

with missing values. The most common approach for missing values is to drop the whole rows including other variables in various attributes. This approach is quite simple and easy, but there is also a risk of deleting important values in other attributes. So, it is necessary to analyse whether the rows to be deleted do not include the important values or not before deleting the whole missing values. Another approach is to fill the missing values with zero [42]. This approach may mislead a machine learning model if the missing values are proceeded incorrectly. For instance, if “0” is already used as a substitution for “no” in an attribute which has missing values, the missing values to be filled with zero will have the meaning of “no” as well. In this case, the missing values need to be filled with an unused number. The other approach is imputation that fills missing data with the median/average of each customer in each attribute.

In any case, the data in each attribute should be analysed before the missing values are dealt with using any one of the approaches.

Eventually, I created the process flow for dealing with missing values correctly as shown in Figure 3-9.

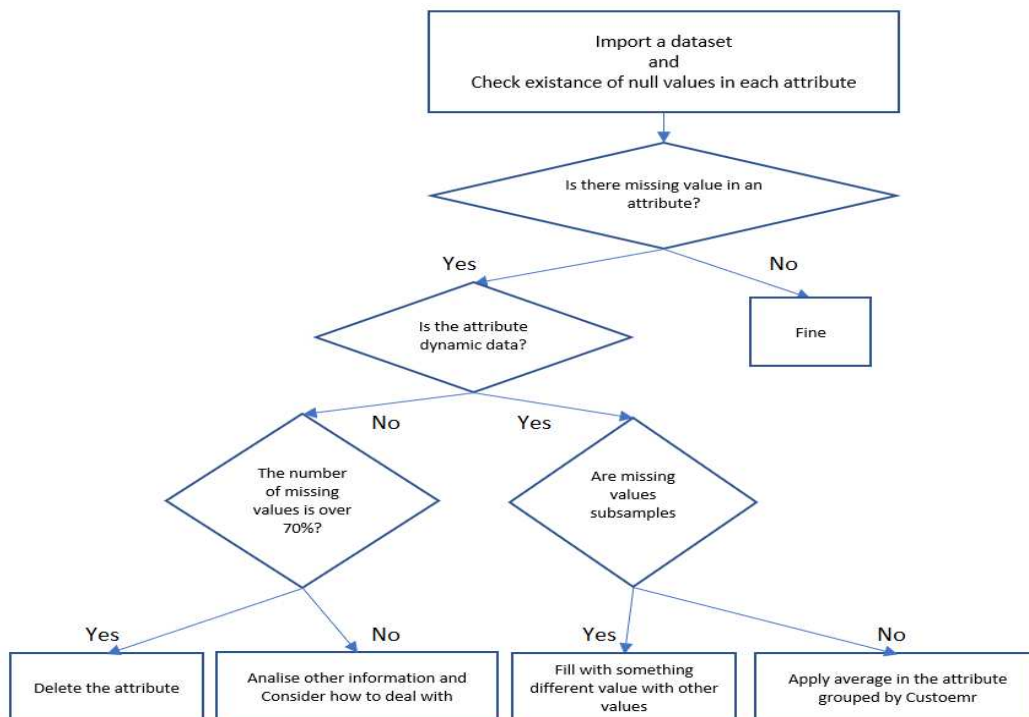


Figure 3-9. A flow of processes for dealing with missing values

### 3.2.2. Feature Creation Processes

The feature creation part is the most significant in the framework because of having an influence over the performance of machine learning models. In any banking system, there are some common data related to transaction such as customer, bank, credit card, timeslot, accessing device and networks. Thus, applicable features which can be extracted from the banking system need to be checked whether they have variables correctly before using for feature aggregation and transformation in the framework.

#### I. Feature Aggregation based on Customer Behaviour

The purpose of creating aggregated features is to have feature values with more clear difference in transaction's behaviour between customer and fraudster. By aggregating feature values of customer's ID and some dynamic features e.g., amount, time, access device and network information, the new aggregated features represent new patterns.

Increasing the related features to a transaction means increasing the dimensions that make a machine learning model with the detailed patterns. Table 3-8 describes some fixed attributes in a transaction and new aggregated features based on a scenario of individual customer's journey via online banking.

Attributes	Feature Aggregation with Scenario of customer's journey on transaction
Time	<ul style="list-style-type: none"> <li>- Days since the last transaction</li> <li>- Hours since the last transaction</li> <li>- Minutes since the last transaction</li> <li>- Weekdays since the last transaction</li> </ul>
Amount	<ul style="list-style-type: none"> <li>- Amount of the last transaction in every month</li> <li>- Average amount of transactions</li> <li>- Count amount of transactions</li> <li>- Last amount of transactions</li> <li>- Maximum amount of transactions</li> <li>- Minimum amount of transactions</li> </ul>
Balance	<ul style="list-style-type: none"> <li>- Balance of the last transaction in every month</li> <li>- Average balance of transactions</li> <li>- Count balance of transactions</li> <li>- Last balance of transactions</li> <li>- Maximum balance of transactions</li> <li>- Minimum balance of transactions</li> </ul>
Login / IP address/ Accessing Device	<ul style="list-style-type: none"> <li>- Last login via a specific IP address</li> <li>- Average Login latency</li> <li>- When accessed with a specific IP address</li> <li>- Which device was used with a specific IP address</li> <li>- Last transaction with a specific IP address</li> <li>- How many accessed with a specific IP address in weekdays/Days</li> </ul>
Event	<ul style="list-style-type: none"> <li>▪ Which event was occurred via a specific device</li> <li>▪ Which event was occurred via a specific financial institution</li> </ul>
Customer ID	<ul style="list-style-type: none"> <li>▪ Count of last transaction per weekdays by a specific IP address based on customer ID</li> <li>▪ Count of last transaction per weekdays and timestamp based on customer ID</li> <li>▪ Count of last transaction per weekdays and last latency based on customer ID</li> <li>▪ Count of last transaction amount by a specific IP address based on customer ID</li> </ul>

Table 3-8. Feature aggregation scenario



## **II. Feature Transformation based on mathematical functions**

As described in Chapter 2, there are existing methods of feature transformation i.e., scaling (standardisation), log transformation, grouping, count, PCA, and statistics, which were used only for transforming a feature value to a different value in various research fields but also used in other research areas [19] [68] [72] [73]. For instance, feature scaling is a method for scaling a wide range of the data variables to the same range and it is popularly applied for deep learning in image recognition. Confidence Interval Formulas were commonly used for observing a point estimate and measuring feature values in data analysis. Logarithm transformation formula were popularly used for removing skewness because machine learning algorithms will be biased when the data distribution is skewed. K-means is used to discover groups which have not been clearly labelled in the data. In general, K-means is not used as a feature engineering method but an unsupervised algorithm that can be used for any type of grouping [92] [106]. However, in terms of generating new feature variables from raw data, it is worth using the output feature variables which were created by the K-means method. The purpose of using feature transformation here is to create new feature values that can represent the latent data patterns and make a machine learning algorithm easily understand the difference between legitimate and fraud. Therefore, in this research the transformation methods are incorporated in the feature engineering framework.

The functions utilised in the framework are outlined below in Table 3-9:

Formula	Description
1) Confidence Interval Formulas	A statistic estimation formula that uses the normal distribution for observing a point estimate by calculating maximum, minimum, median, and mean.
2) Standard Deviation	A method of scaling the values based on z-score which calculates the following equation: $\text{Standard Deviation} = \sqrt{\frac{\sum_{i=1}^n (X_i - \bar{X})^2}{N-1}}$
3) Logarithm transformation formula	A method of removing skewness by adapting the formula below: $x'_i = \log(x_i)$
4) K-Means clustering	The objective of k-means is to minimize the squared error function. The objective function is as follows: $J = \sum_{i=1}^m \sum_{k=1}^K w_{ik} \ x^i - \mu_k\ ^2$
5) Principal Component Analysis (PCA)	A method of exploring correlations among the given feature values and integrating into a similar group as a new feature. $V_{xx} = \frac{1}{N} \sum_{i=1}^N (X^i - \frac{1}{N} \sum_{j=1}^N X^j)(X^i - \frac{1}{N} \sum_{j=1}^N X^j)^T$ $\max_{\ a\ =1} \frac{1}{N} \sum_{i=1}^N (a^T (X^i - \frac{1}{N} \sum_{j=1}^N X^j))^2 = \max_{\ a\ =1} a^T V_{xx} a$

Table 3-9. Feature transformation methods in the research

In terms of dealing with only PCA, this transformation needs to be implemented after standardisation because PCA provides more weights to the variables in a selected feature by measuring each variable equally.

### III. Standardisation

Variables in each feature that are measured at different scales do not contribute equally to the ML/DL model fitting. Standardisation is an essential data processing specifically for using deep learning because deep learning calculates the given feature values by multiplying each other for determining the weighting coefficients in hidden layers. It is also important for non-tree machine learning models, such as support vector machine (SVM), clustering models, are often hugely dependent on scaling (this sentence needs

fixing). For instance, SVM attempts to maximise the distance between the boundary line and the support vectors. If one feature has very vast values, it will monopolise over other features during distance measurement. Thus, feature-wise standardised ( $\mu = 0, \sigma = 1$ ) is usually used prior to model fitting for the purpose of giving the same influence on the measured distance.

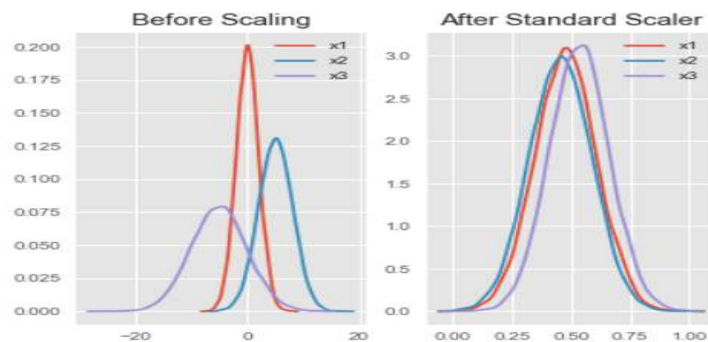


Figure 3-10. Standardisation feature values [98]

Standardisation is a step of data pre-processing which is applied to independent variables in features. In Figure 3-10, before standardizing variables in each feature of x1,x2 and x3, the range of data in each distribution is different. This will make a non-tree machine learning model which uses Euclidean Distance measure mislead measurement of a target. On the other hand, after standardizing variables in each feature of x1,x2 and x3, the range of data in each distribution was common and can be compared and equally measured to weight all the features. It also will help in speeding up the calculations in the ML models. The data values standard in each feature are rescaled by calculating the following formula:

$$\text{Standardisation} = \frac{X - \text{mean}(X)}{\text{Standard Deviation}(X)} \quad (\text{eq. 3.4})$$

To standardise the data, we use the StandardScaler from the sklearn python library:

```
from sklearn.preprocessing import StandardScaler
X_train_stand = X_train.copy()
X_test_stand = X_test.copy()
scale = StandardScaler().fit(X_train_stand[[i]])
X_train_stand[i] = scale.transform(X_train_stand[[i]])
X_test_stand[i] = scale.transform(X_test_stand[[i]])
```

With the feature creation processes which apply many feature engineering techniques on an original dataset, a large number of features are generated. However, all features may not be useful for machine learning algorithms. Now I will consider feature selection processes to reduce meaningless or high correlation features.

### **3.3. Feature Selection Process Component**

Through the processes in the feature creation component, many new features were added in the dataset. However, redundant features might be also included in the dataset, which have an impact on model performance for the worse and will cause overfitting [58]. Under the component of feature selection processes, there are two main parts to create new features: Feature Selection and Performance Measurement.

Feature selection is a process of selecting the most useful features to train on among input variables. If the training data contains too many irrelevant features, the performance of machine learning models may degrade. In the feature selection section, there are three main steps for selecting the effective features from all features in the dataset including new features: Measurement for correlation coefficient values, building a feature selection model, and feature importance measurement. In the feature measurement part, I calculate correlation coefficient and measure feature importance, and then drop redundant features.

Another method for feature selection is the performance measurement which consists of modelling with three different types of feature sets and evaluating the model performance.

### 3.3.1. Feature Selection

#### I. Measurement for correlation coefficient values

When there are high correlations between two or more explanatory variables in the dataset, multicollinearity exists and will cause overfitting in a multiple regression model. The Pearson correlation coefficient is a statistical method to measure the degree of intensity of the relationship between two random feature variables, X and Y [107].

In the framework, I selected Pearson correlation to calculate the strength between two variables from different types of correlation coefficients. The range of the strength values of the correlation is expressed between -1 and 1. A value of -1 indicates the perfect negative relationship between the two feature values. On the contrary, a value of 1 indicates the perfect positive relationship between the two feature values. Values close to zero means weak or no relationship between the two values as shown in Table 3-10. The equation of the Pearson correlation coefficient is shown below with a correlation of a variable of 1 [107]:

$$\rho_{xy} = \frac{Cov(x,y)}{\sigma_x\sigma_y} = 1 \quad (\text{eq. 3.5})$$

Where:

$\rho_{xy}$  = *Pearson product – moment correlation coefficient*

$Cov(x, y)$  = *Covariance of variables x and y*

$\sigma_x$  = *Standard deviation of x*

$\sigma_y$  = *Standard deviation of y*

Range of Correlation	Interpretation
+/- 0.9 to +/- 1.0	Very high positive(negative) correlation
+/- 0.7 to +/- 0.9	High positive (negative) correlation
+/- 0.5 to +/- 0.7	Moderate positive (negative) correlation
+/-0.1 to +/- 0.5	Low positive (negative) correlation
0.0	No correlation

Table 3-10. Benchmark of correlation coefficient

In the experiment, the Pearson correlation coefficient is computed in python using the corr() method as shown in Figure 3-11.

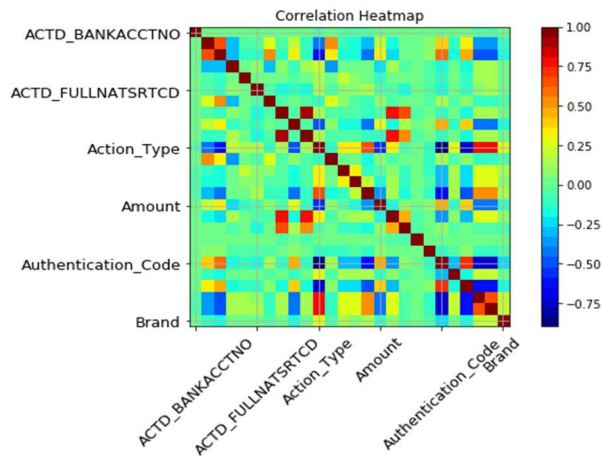


Figure 3-11. Sample of correlation matrix

Since the dataset is too large to display all results between every pair of features, as a sample of correlations between a specific feature and others, Table 3-11 shows how much each feature correlates with the bank account number:

```
Corr_matrix= df_banking_dataset.corr()
Corr_matrix[“ACTD_BANKACCTNO”].sort_values(ascending=False)
```

	<b>ACTD_BANKACCTNO</b>
ACTD_BANKACCTNO	1
ACTD_ACCTTYPENM	0.645984
LATENCY_log	0.397597
LATENCY_std	0.235693
Event_INTERNET_BANKING	0.064752
Authentication_Code_log	0.061193
Class	0.031434
TRNSD_FASTERSTANDARDPAYMENTIND	0.02844
ACTD_AVAILABLEBL	0.017514
Transaction_ID	0.015699
Latency_diff	0.010944
Event_DIGITAL BRANCH	-0.041485
Amount_std	-0.062837
IP_ID	-0.065832
Authentication_Code	-0.067865
Channel	-0.069975
DEVICE	-0.079998
Weekday	-0.114141
Fraud_flag	-0.135582
IDVD_USAGE0TTX	-0.140925
Event_INC_Code	-0.143715
Amount_confRate	-0.147124
Subchannel	-0.156701
Event_Sub_Device	-0.169679
Customer_ID_confRate	-0.190379
IP_Adress_confRate	-0.206447
CUSTD_PARTYID_TRNSD_Amount_log_count _LATUPDATE_Weekdays	-0.213371
ACTD_AVAILABLEBLBALANCE_min_mean	-0.256167
ACTD_AVAILABLEBLBALANCE_log	-0.264038
IDVD_IPADDRESSID	-0.283122
IP_Adress_log	-0.310436
Action_Type	-0.311109
IDVDATA_TRNSTS	-0.315315
Channel_Event	-0.383213
Event_Act	-0.407158
Action_Type_confRate	-0.419303
Brand	-0.43251

*Table 3-11. Correlations with bank account number*

Note that the correlation coefficient only measures linear correlations, which means that if  $x$  goes up, then  $y$  normally goes up or down. It is not directly for measuring nonlinear relationships. However, it is important to understand the correlation coefficient with each attribute because the high correlated values may cause of leakage or overfitting in machine learning.

In my framework, the threshold value to drop either two variables which have high correlation coefficient is set above  $\pm 0.9$ .

## **II. Feature Importance Measurement**

Feature importance scores can give an insight into the dataset. The relative scores can emphasize which attributes may be relevant to the target, and which attributes are the least relevant. To calculate a feature importance score in each attribute, I built a random forest algorithm which is an ensemble of decision trees. A random number of rows and all the attributes from the dataset are selected, which is known as bootstrapping, and then some features will be randomly selected and start building decision trees. In the algorithm, it will build multiple decision trees in parallel based on information obtained by subtracting entropy or Gini index and calculating node impurities of each of the appropriate attribute where it is branching. Each decision tree will become larger to its maximum depth and will provide prediction. The outline processes in the random forest algorithm are shown in the below Figure 3-12.



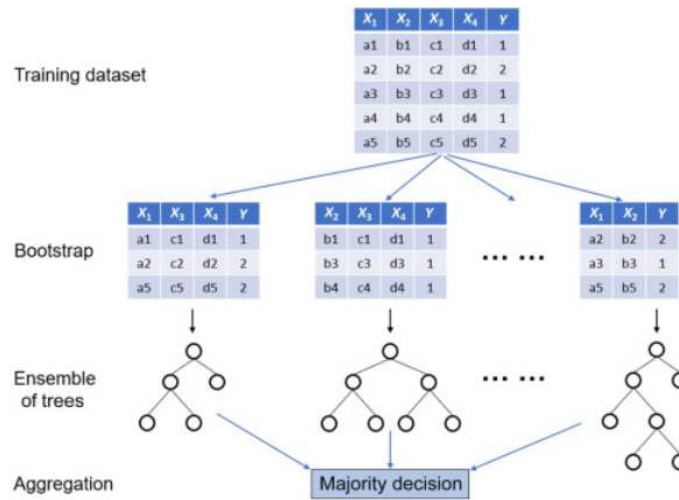


Figure 3-12. Outline processes in a random forest algorithm [100]

Figure 3-13 shows the example of two decision trees in ensemble of trees which calculates node impurities from if that appropriate attribute is branching out.

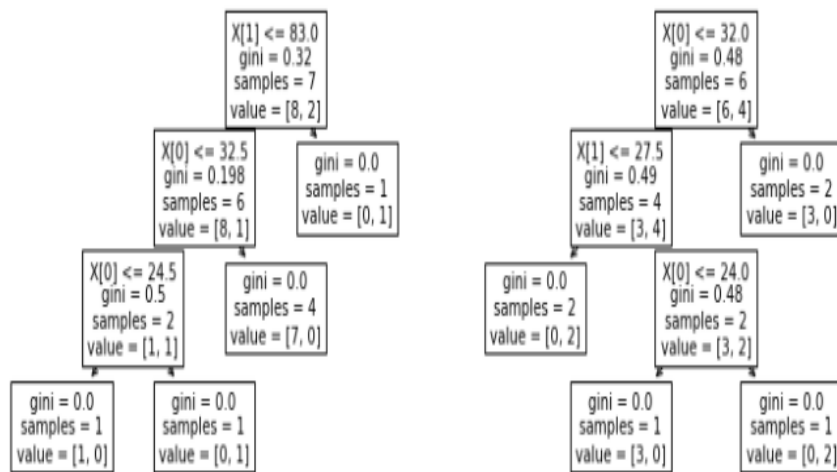


Figure 3-13. Decision trees inside of random forests [59]

The node impurity can be gained from a Random Forest algorithm using the following equation [59]:

$$Node\ Impurity = \left( \left( \frac{N_t}{N_p} \right) * G_i \right) - \left( \left( \frac{N_{tr}}{N_t} \right) * G_{ir} \right) - \left( \left( \frac{N_{tl}}{N_t} \right) * G_{il} \right) \quad (eq. 3.6)$$

Where;

$N_t$  = number of samples for the appropriate node

$N_P$  = number of samples chosen at the previous node

$N_{tr}$  = number of samples branched out in the right node from main node

$N_{tl}$  = number of samples branched out in the left node from main node

$G_i$  = Gini index of the appropriate node

$G_{tr}$  = Gini index of the right node branching from main node

$G_{tl}$  = Gini index of the left node branching from main node

In order to obtain the above feature importance scores, a feature selection model with the random forest algorithm is built through using the Python library of the RandomForestClassifier class in scikit-learn which is convenient and optimised for decision trees . Further details on the RF model will be covered in Chapter 5.

The following code is available in the Python library which exhibit feature importance scores.

```
from sklearn.ensemble import RandomForestClassifier
df_rf = RandomForestClassifier (n_estimators=500, max_leaf_nodes=16, n_jobs=-1)
df_rf.fit (X_train, y_train)
y_pred_rf = df_rf.predict (X_test)
importance = df_rf.feature_importances_
```

The graph below shows an example of the best 30 feature importance scores in ascending order in Figure 3-14.

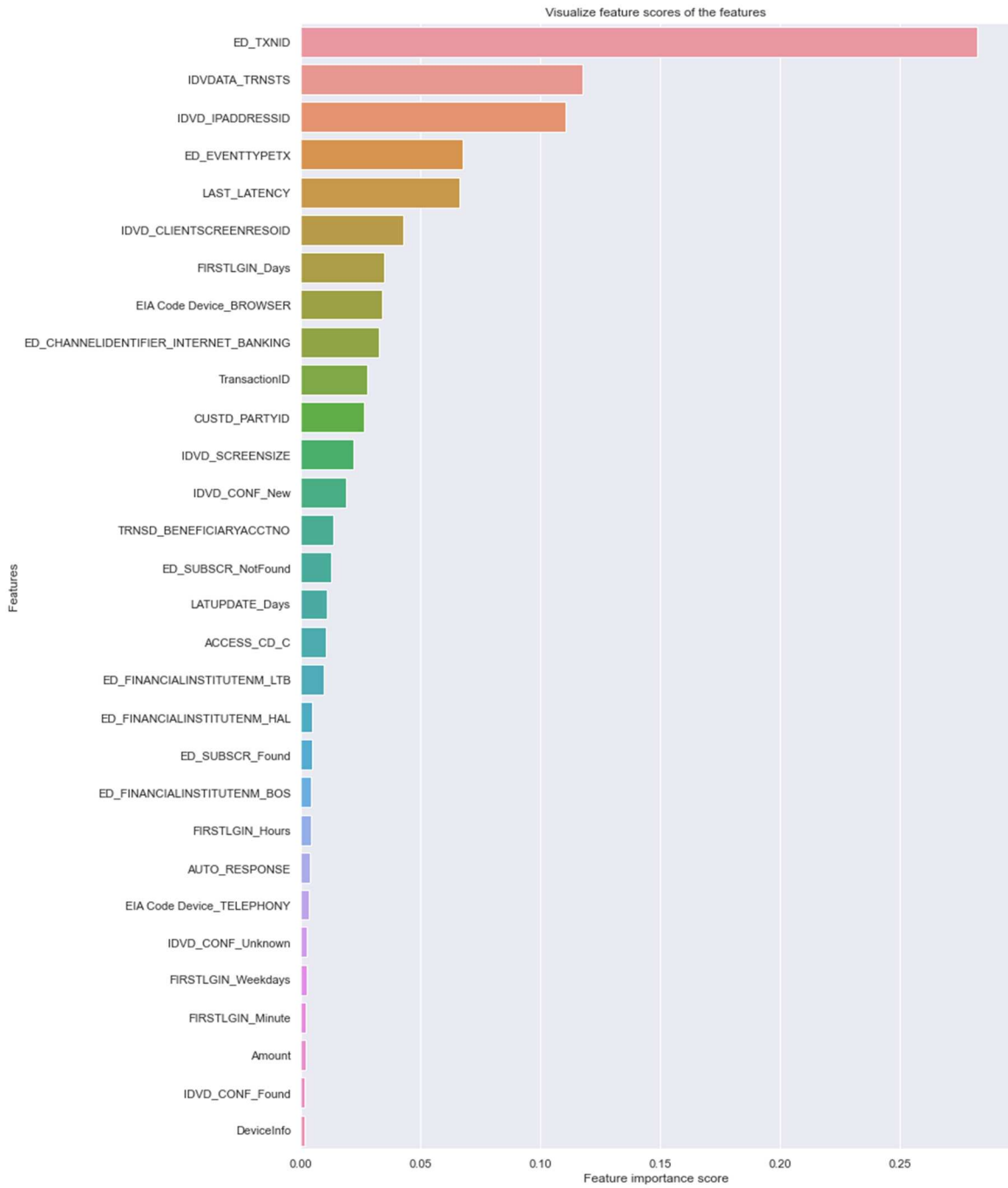


Figure 3-14. Feature importance scores of top 30 features

In my framework, based on the result of feature importance scores, features with 0.0 score are dropped from the dataset.

### **3.3.2. Performance Metrics for Fraud Detection Models**

#### **I. Modelling**

To evaluate the effectiveness of the created feature sets, several combinations of machine learning and deep learning algorithms and the different features sets are within each model. As introduced in many studies pertinent to fraud detection (in Chapter 2), a variety of algorithms were used as a fraud detection model in their studies. These algorithms had individual methods to deal with the input feature values and expressed various performances. To secure consistency the effectiveness of created feature set, various algorithms which were often used in a fraud detection case should be tested with the features sets created in the framework and their performances need to be compared.

In the experiment, I use support vector machine, random forest, isolation forest, local outlier factor, and autoencoder for model verification. All algorithms are popularly used in the studies for fraud detection. One of my framework's strengths is that it can provide the most effective features set for a specific machine learning or deep learning algorithm which one wants to use. The framework is flexible and adaptable for any models and can provide the best combination features set.

Here are the recaps of using these algorithms in the experiment:

#### **(A) Support Vector Machines Classification**

SVM is one of the versatile machine learning model, capable of performing linear or nonlinear classification. It is particularly well suited for classification of the medium-sized dataset. Although linear SVM classifiers are capable and work well in many cases, lots of datasets will not be close to being linearly separable. Nonlinear SVM can handle

the datasets that added more features and are not linearly separable as shown in Figure 3-15.

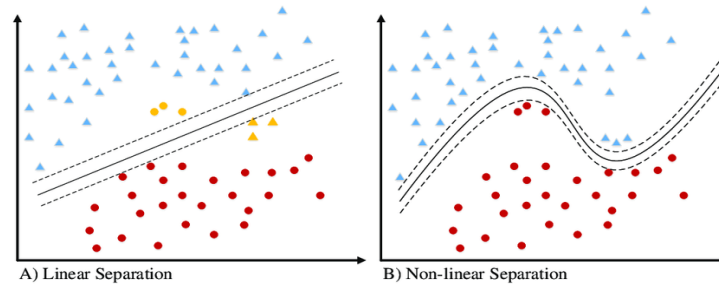


Figure 3-15. Linear and nonlinear separation of sample data [99]

In order to tackle the nonlinear separation datasets, there is a mathematical technique called the kernel trick that is used for bridging linear and nonlinear SVM. Figure 3-16 describes the way to map samples from two-dimensional space to three-dimensional space by using a kernel function. It shows a decision surface that apparently divides between different classes.

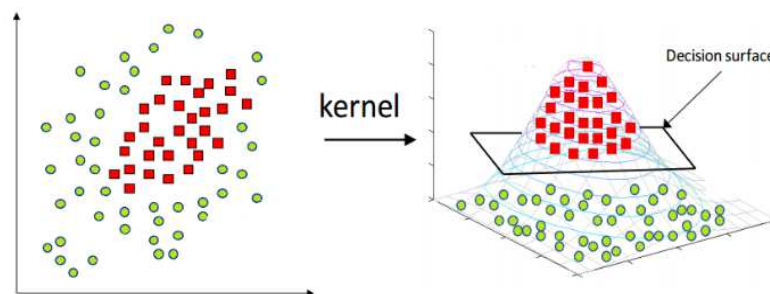


Figure 3-16. Mapping the data from two-dimensional space to three-dimensional space [88]

The kernel trick can provide a more efficient and simple way to transform data into higher dimensions. There are two kernels that are popularly used in SVM classifier, the polynomial kernel and the radial basis function (RBF) kernel. The polynomial kernel uses the following mathematical function to map the data to higher dimensions [101]:

$$\phi(x, y) = (X^T y + 1)^d \quad (\text{eq. 3.7})$$

With  $N$  original features and  $d$  degrees of polynomial, the polynomial kernel outputs  $N^d$  expanded features. Figure 3-17 shows the SVM classifier using a 3<sup>rd</sup>-degree polynomial kernel. If a SVM model is overfitting, the number of degree can be reduced. Conversely, if the model is underfitting, the number of degree can be increased as well. Finding out the right hyperparameter will be key to become a good model.

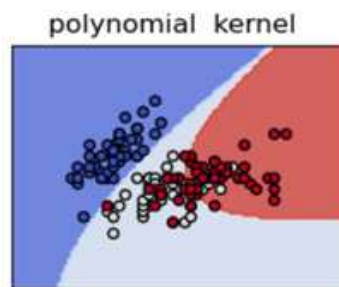


Figure 3-17. Nonlinear SVM with polynomial kernel [101]

The RBF kernel is to add features calculated using a similarity function, which adjusts how much each sample resembles a particular landmark, and it is defined the bell-shaped function. The following function is used for mapping the data to higher dimensions [101]:

$$\emptyset(X, Y) = \exp(-\gamma \|X - Y\|^2) \quad (\text{eq. 3.8})$$

The parameter of  $\gamma$  determines how much influence a single sample has. Figure 3-18 shows the plot of the SVM classifier using an RBF kernel.

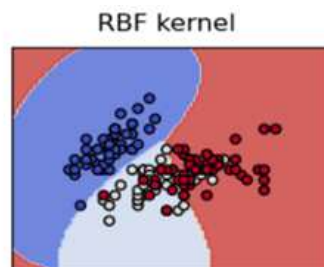


Figure 3-18. SVM classifier using an RBF kernel [101]

A common approach to adjusting the hyperparameter is to use grid search in practical terms because of its difficulty to determine an appropriate parameter manually. However, in Python library, it provides a package of the SVM algorithm with some default setup.

### **(B) One-class Support Vector Machine (SVM)**

The above SVM classifiers are general methods used for classification. A one-class SVM is provided for outlier detection, and it is better suited for fraud detection [72]. The one-class SVM algorithm instead tries to isolate the cases in high-dimensional space from the origin. In the original space, this will conform to discovering a small region that covers all the cases. If a new case does not belong with this region, it is a fraud. It determines a smooth boundary that separates the transformed vectors into normal and anomaly samples, and then builds up a model of a normal behaviour, where points of data that depart from that model are classified as anomalies. The popular function in the one-class SVM is support vector data description (SVDD) which was introduced by Tax and Duin in 2004. The concept of SVDD is that a minimum radius hypersphere is fixed around most of the transformed vectors in the feature space. The data points that settle outside the hypersphere are classified as anomalies as shown in Figure 3-19 [89].

Given training data:  $S = \{x_1, \dots, x_l\}, x_i \in \mathbb{R}^n$ ,

$\xi_1$  is the slack variables

$v$ : the regularisation parameter

$\varphi(\blacksquare)$ : the kernel function

$R$ : the radius,  $a$ : the centre of the hypersphere

$$\min_{\alpha, R, \xi} R^2 + \frac{1}{mv} \sum_l^m \xi_1 \quad (\text{eq. 3.9})$$

such that

$$\|\phi(x_l) - \alpha\| \leq R^2 + \xi_1, \forall l = 1, \dots, m, \xi_1 \geq 0 \quad (\text{eq. 3.10})$$

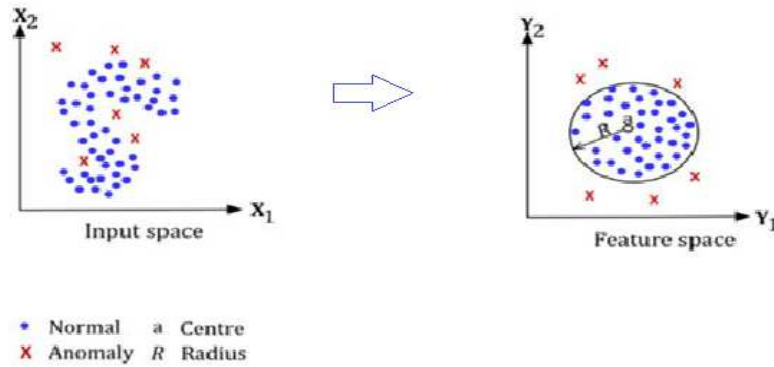


Figure 3-19. One-class support vector machine [89]

As well as nonlinear SVM classification above, the one-class SVM method is also provided the package from scikit-learn . There are a few hyperparameters to fine-tune, which works good especially with high-dimensional datasets.

### (C) Random Forest (RF)

Random forest is a supervised learning algorithm and consists of many decision trees. It is a highly accurate and robust method because of the number of decision trees performing in the process. The algorithm works as shown in Figure 3-20 and proceeds with the following steps:

- 1) Randomly select samples from a given dataset
- 2) Build up a decision tree for each sample and gain a prediction result from each decision tree.
- 3) Carry out a vote for each predicted result
- 4) Select the prediction result with the most votes as the final prediction



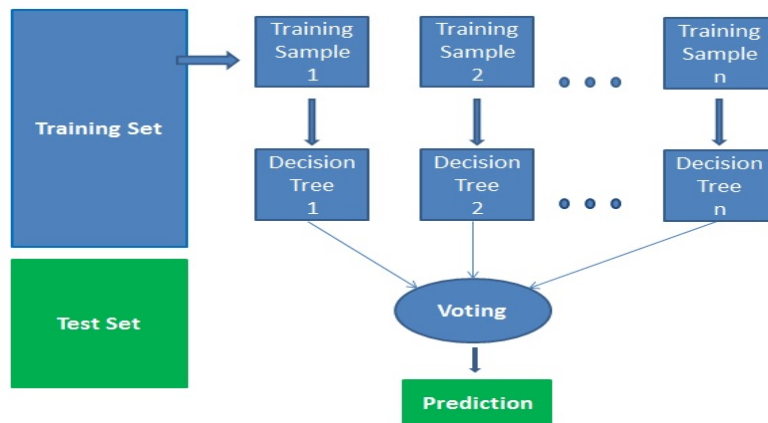


Figure 3-20. The steps of Random Forest algorithm [107]

### (D) Isolation Forest (IF)

This is an efficient algorithm for anomaly detection, especially in high-dimensional datasets and builds a random forest where each decision tree is grown randomly: it selects a feature randomly at individual node, then it selects a value of threshold randomly from between the minimum and maximum values for splitting the dataset into two. It compares the density of samples around a given sample to the density around its neighbours. An anomaly is regularly more isolated than its  $k$  nearest neighbours. To observe anomalies, the processes are as follows:

- 1) Choose a feature randomly and choose a value for that feature randomly within its spectrum.
- 2) The value becomes the new maximum (minimum) of that feature's spectrum if the sample's feature value falls below (above) the chosen value.
- 3) Confirm if at least one other sample has values in the spectrum of each feature in the dataset, where some spectrums were adjusted via step2. If not, then the sample is isolated.

- 4) Reiterate step1 to 3 until the sample is isolated. The number of times I go through the above steps is the isolation number. The lower the number, the more abnormal the sample is.

**(E) Local Outlier Factor (LOF)**

Local outlier factor calculates the local density deviation of a certain data point in terms of the number of its neighbours and compares it to the density of other points. The higher the local outlier factor value for a samples, the more fraudulent the sample. In the below image of feature space, LOF can identify T1,T2 and T3 as frauds, which are local outliers to Cluster 2 in Figure 3-21.

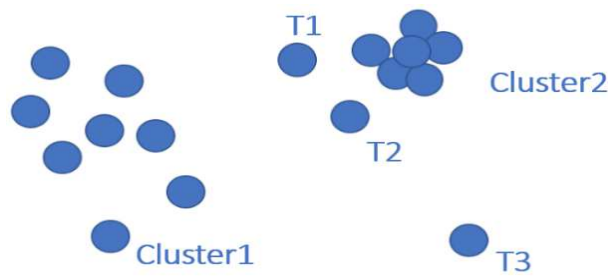


Figure 3-21. Instance of LOF [102]

To define each data point, the following processes will be carried out [102]:

- 1) Compute distances between a certain data point (P) and every other point by using Manhattan distance which is a distance formula between two points in a certain dimensional vector space.

$$\|x_1 - x_2\| + \|y_1 - y_2\| = dist(p1, p2) \tag{eq. 3.11}$$

- 2) Discover the Kth closest point:

$$Kth\ nearest\ neighbour's\ distance = K - Dist(P) \tag{eq. 3.12}$$

- 3) Discover the  $K$  closest points, where those whose distances are smaller than the  $K$ th point. The  $K$ -distance neighbourhood of  $P$ ,  $N_k(P)$ .
- 4) Discover its density by assessing how close its neighbours are to it. Generally, the inverse of the average distance between point  $P$  and its neighbours. The lower the density, the farther  $P$  is from its neighbours.

$$\text{Local reachability density} = \text{LRD}_k(P)$$

- 5) Discover its local outlier factor ( $\text{LOF}_k(P)$ ):  $\text{LOF}_k(P)$  is fundamentally the sum of the distance between  $P$  and its neighbours, weighted by the sum those point's densities.

#### **(F) Autoencoder (AE)**

Autoencoder is a typical deep neural networks among others and shares a strong resemblance with multilayer perceptron neurons and then an output layer. AE is artificial neural networks capable of learning dense representations of the input data. As an architecture of the autoencoder, hyperparameter that is the number of nodes in the code layer needs to be determined before training the AE. Figure 3-22 describes the autoencoder architecture which consists of two parts: an encoder and a decoder. The encoder converts the input dataset of features into a different representation whereas the decoder converts this freshly learned representation to the original layout. The encoder part should be taken care of most because the new features set can be derived from the original dataset of features. I will refer to the encoder function of the autoencoder as  $h=f(x)$ , which takes in the original observations  $x$  and utilises the freshly learned representation captured in function  $f$  to output  $h$ . The decoder function that reconstructs the original observations utilising the encoder function is  $r=g(h)$ .

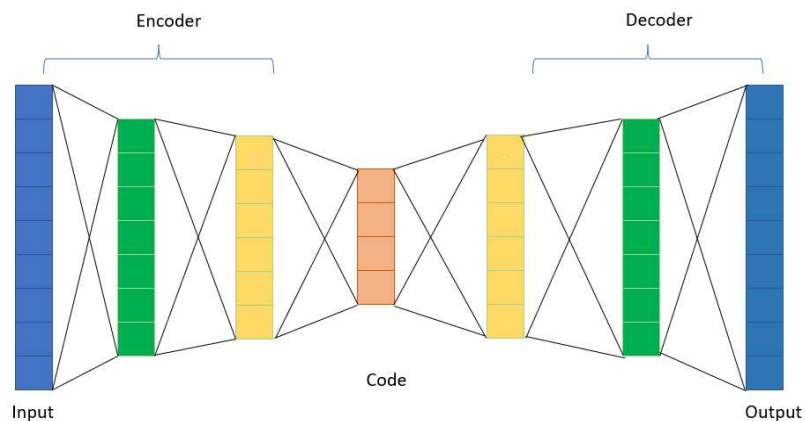


Figure 3-22. Architecture of autoencoder [52]

Before training the autoencoder, I need to grasp some core components in neural networks and determine which functions and parameters are used for modelling as presented below:

- Active Functions

An activation function is applied to the hidden layer to reconstruct the original observations. This activation functions represents the decoder portion of the autoencoder. The output layer represents the newly reconstructed observations. To compute the reconstruction error, I will compare the newly constructed observations with the original ones. A neural network learns the weights to apply to the nodes at each of the layers but whether the nodes will be activated or not is decided by the activation function. Specifically, an activation function is applied to the weighted input and bias at each layer. The information in each node is passed to the next layer. However, it is not simply binary activations. In order to have a range of activation values, I can select a linear activation function or a nonlinear activation function. The linear activation function is unbounded. It can produce activation values between negative infinity and positive infinity. Popular nonlinear activation functions have sigmoid, tanh, rectified linear unit (ReLU), exponential linear unit (ELU), and softmax. The sigmoid function is bounded and can produce activation values between

0 and 1. The tanh function is also bounded and can produce activation values between -1 and +1. Its gradient is steeper than that of the sigmoid function. The ReLu function has an interesting property. If the weighted input plus bias is positive, ReLu will return the weighted input plus bias. Otherwise, it will return zero. Thus, ReLu is unbounded for positive values of the weighted input plus bias. The ELU function is unbounded and adapts a log curve for negative values. It is based on ReLu that has an extra alpha constant  $\alpha$  except negative points. The ELU is also in identity function form for nonnegative inputs, but it becomes smooth gradually until its output equal to  $-\alpha$ . It tends to converge faster than ReLu. Lastly, the softmax function is utilised as the final activation function in a neural network for classification problems because it normalises classification probabilities to values that aggregate to a probability of one.

Among all these functions, the linear activation function is the simplest and least computationally expensive. ELU and ReLu are the most popular activation functions used. ReLu is the next least computationally expensive followed by others. The ELU function is better generalisation performance than ReLu and fully continuous.

- Number of Hidden Layers

The number of hidden layers plus the output layer count toward the number of layers in a neural network. As I learned in Chapter 2, layers will be represented by subnetworks in certain architectures.

- Number of Epochs/Optimizer

Neural networks train for many rounds, which is known as epochs. In each of these epochs, the neural network adapts its learned weights to lower its loss from the previous epoch. The number of epochs determines the number of times the training

occurs over the full dataset passed into the neural network. The process for learning the weights is fixed by the optimizer. That process boosts the neural network to efficiently learn the optimal weights for the various nodes across whole layers which minimizes the loss function I have selected. The neural network requires to adjust its predictions for the optimal weights in an intelligent way to learn. One approach is to iteratively move the weights in the direction that boosts lower the loss function increasingly. However, an even better approach is to move the weights in this direction but with a degree of randomness. This process is known as stochastic gradient descent (SGD). Generally, optimizer is used in training neural networks the most. SGD has a single learning rate for all the weight updates that it makes, and this learning rate does not change during training. However, in most studies, it's better to adapt the learning rate over the circuit of the training. For instance, in the earlier epochs, it becomes more reasonable to adapt the weights by a large degree and to have a large learning rate. Conversely, in later epochs, when the weights are more optimal, it becomes more reasonable to adapt the weights by a small degree to delicately fine-tune the weights than to take massive processes in one direction or another. Thus, the Adam optimization algorithm which is derived from adaptive moment estimation is a better optimizer than SGD. The Adam optimizer dynamically adapts the learning rate over the circuit of the training process, different from SGD.

- Loss Function

Regarding the loss function, in order to evaluate the model based on the reconstruction error between the newly reconstructed matrix of features based on the autoencoder and the original feature matrix that I feed into the autoencoder, the evaluation metric needs to be selected.

- Batch Size

The batch sets the number of samples the neural network trains on before creating the next gradient update. If the batch is equal to the total number of observations, the neural network will make a gradient update once every epoch. Alternatively, it will create updates multiple times per epoch.

The detailed implementation is described in Chapter 5. I used the Scikit-Learn which provides the autoencoder library to build the AE model.

## **II. Performance Evaluation**

The most common metrics of model validation is accuracy, the area under the receiver operating characteristic curve (AUC), true positive (TP) and false negative (FN). Accuracy is derived by the ratio of number of correct predictions from the total number of observation samples. However, evaluating the model performance in the case of dealing with unbalanced labelled dataset by only accuracy is not precise because there is possibility of being a good accuracy score despite of only correcting a majority class which is nonfraud and failing in detecting a minority class which is fraud. I need to build and compare the model performance with appropriate metrics for considering a balance between the true positives ratio (TPR) and the false-positive ratio (FPR). True positive (TP) is the number of predictions as fraud where the actual result is also fraud. On the other hand, false positive (FP) is the number of predictions as a legitimate transaction where the actual result is the customer. True negatives (TN) and false negatives (FN) are also significant metrics when measuring the performance of recall and precision. TN is the number of predictions of fraud where actual result was also fraud. FN is the number

of predictions of fraud where actual result was nonfraud. A summary of the TP, FP, TN, and FN, which is called confusion matrix is shown in Table 3-12.

# of observations	Actual Nonfraud	Actual Fraud
Predicted Nonfraud	True Positives (TP)	False Positives (FP)
Predicted Fraud	False Negatives (FN)	True Negatives (TN)

*Table 3-12. Confusion matrix*

For the imbalanced transactions dataset, a better way to evaluate the performance of models is to apply recall and precision. Recall is the number of true positives over the number of total actual positives in the dataset. For fraud detection, recall measures the number of fraud detections made from all fraudulent samples in the dataset. A high recall indicates that the model has detected most of the true frauds. On the other hand, precision is the number of true positives over the number of total positive predictions. For fraud detection, precision assesses the number of fraud detections that truly belong to frauds. A high precision indicates that many true frauds are detected from all of positive predictions. In the high precision and low recall case, the financial institution would lose lots of money because of fraud, however, it would not cause problem on customers by unwanted rejecting transactions. In the low precision and high recall case, the financial institution would detect lots of the fraud, however, it would most absolutely anger customers by unwanted rejecting lots of the legitimated transactions by customers. It is a trade-off between recall and precision.



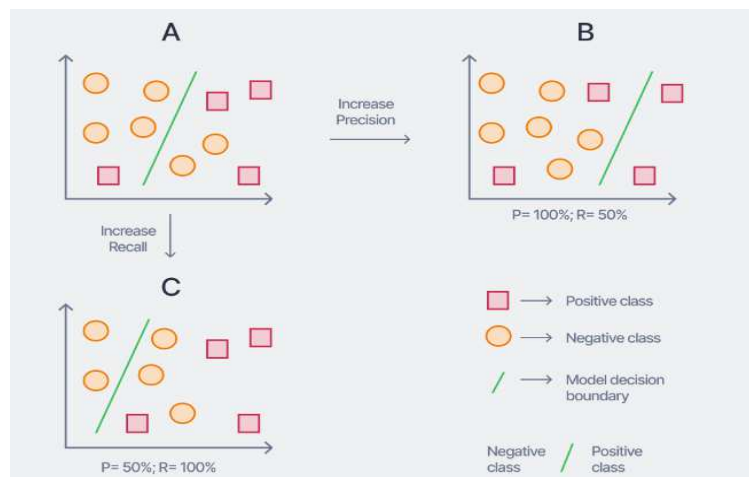


Figure 3-23. Precision versus Recall [109]

As shown in Figure 3-23, the first scenario A consists of positive class samples which is in red and negative class samples highlighted in yellow. The green line is the decision boundary which divides the samples into the positive class and negative class. For instance, to increase precision, the green threshold is shifted to the right-hand side and became scenario B. Precision becomes 100% as positive samples on the right-hand side divided by total samples on right side makes  $2/2$ . Then, recall becomes 50% in scenario B because positive samples on right side divided by total positive samples makes  $2/4$ . When precision increases, recall decreases. On the other hand, to increase recall, the green threshold is changed in the left-hand side and became scenario C. Recall becomes 100% because positive samples on right side divided by total positive samples makes  $4/4$ . In this case, precision became lower than recall score. This scenario describes that any of the positive samples will not be missed although many negative samples will be allowed to get on the right side. In the example, only two thresholds were determined. However, precision and recall scores are calculated across many thresholds and these scores will draw a curve with precision as the y-axis and recall as the x-axis as shown in Figure 3-24 [110].

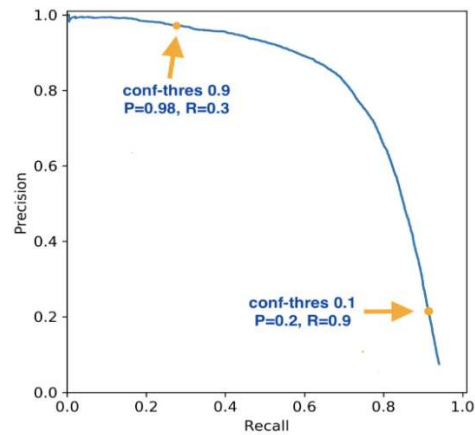


Figure 3-24. Example of a precision and recall curve [110]

The above figure shows a trade-off between false positives and false negatives. F1-Measure is the harmonic mean of precision and recall. The best score is 1 whereas the worst score is 0. This metric seeks the balance between precision and recall as shown in Table 3-13.

Precision	$\frac{TP}{TP + FN}$
Recall	$\frac{TP}{TP + FP}$
F1-measure	$2 * \frac{Precision * Recall}{Precision + Recall}$

Table 3-13. Performance metrics definition

One method to compute classifiers is to measure AUC which stands for “Area under the ROC Curve.” AUC is a performance measurement for the classification issues at different threshold settings. Another common method is ROC curve which stands for “Receiver Operating Characteristic Curve” and draws two parameters: the true positive rate (TPR) and the false positive rate (FPR). TPR is an equivalent with recall or specificity, which is the ratio of negative instances that are precisely classified as negative. The FRP is the ratio of negative instances that are wrongly classifies as positive. TPR and FPR are defined as follows:

$$TPR = \frac{TP}{TP+FN} \quad (\text{eq. 3.13})$$

$$FPR = \frac{FP}{FP+FN} \quad (\text{eq. 3.14})$$

An ROC curve draws TPR against FPR at various classification thresholds. There is a trade-off: The higher the recall (TPR), the more false positives (FPR) the classifier produces.

Figure 3-25 shows a general ROC curve.

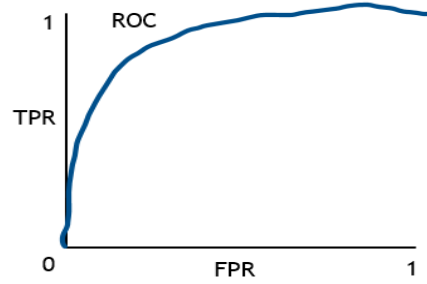


Figure 3-25. Typical ROC curve

AUC describes the measurement of separability which shows how much the model is capable of dividing between classes and adjusts the whole two-dimensional area underneath the whole ROC curve from 0 to 1 as shown in Figure 3-26.

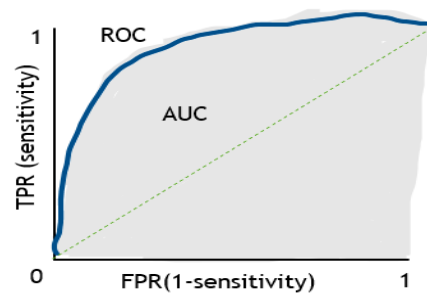


Figure 3-26. Area under the ROC curve (AUC)

The better model is at predicting 0 classes as 0 and 1 classes as 1. AUC ranges in value from 0 to 1. If a model could predict 100% correct, then AUC range becomes 1.0 in the

higher area than the threshold. If a model predicted 100% wrong, then AUC range becomes 0.0 in lower area than the threshold.

Scikit-Learn provides the precision, recall, f1-measure, and AUC functions as below:

```
from sklearn.metrics import precision_score, recall_score, f1_score, roc_auc_score
precision_score (Train_y, Test_y)
recall_score (Train_y, Test_y)
f1_score (Train_y, Test_y)
roc_auc_score (Train_y, y_scores)
```

## 3.4. Key Summary

This section describes the high-level summary of the content of this chapter.

- Proposed a new feature engineering framework that can generate an effective feature set by using various feature engineering methods and selecting practical features from all attributes.
- Made a suggestion of using a flow of processes for dealing with missing values appropriately and presented various options for dealing with categorical values.
- Presented details data cleaning techniques for preparing an appropriate data before feature engineering.
- Provided a technique to perform feature aggregation and transformation is discussed. The focus is specifically on creating features that are based on customer's behaviour on transactions and shows new aspects of input values and enable a ML/DL algorithm easy to learn the difference between normal and fraudulent behaviour.
- Introduced concepts of what the new feature engineering framework is and justified why feature selection is necessary to be embedded in the framework; also proposed techniques on how to estimate the feature values.
- Provided an insight into the use of machine learning and deep learning models for fraud detection and how they can deal with the input data inside of their algorithms.
- Techniques to evaluate the effectiveness of the prepared feature set from the framework are appropriately discussed and proposed.

## **4. Online Banking Transaction Data**

### **4.1. Data Source and Description**

The online banking datasets are provided by a European private bank for only academic purposes and are anonymised. They contain about 130,000 transactions including fraudulent actions which account for 5 % of all transaction that are conducted via the online banking and were collected from 7<sup>th</sup> September 2015 to 7<sup>th</sup> July 2016. The original data was comprised over 300 JSON files assembled from 7 different data tables which are linking to the related tables in an online banking system as shown in Figure 4-1. Each file contains the history data of transactions by many customers conducted during the time period from September 2015 to July 2016. Generally, it is not difficult to read a JSON format file by using a Python library, however, the JSON files I obtained were composed of a very complicated structures and simply could not be loaded under the python environment in my computer. In order to read the whole files and convert to CSV format, I used a free open software which enables me to convert the JSON files into CSV files without using python. Although it worked and I managed to convert the JSON files into the CSV files, one difficulty to use this software was the time it took (time-consuming). I had to load the files one by one manually because the software could not read and proceed multiple files at one time. After reading all JSON files, the same number of CSV files were generated.

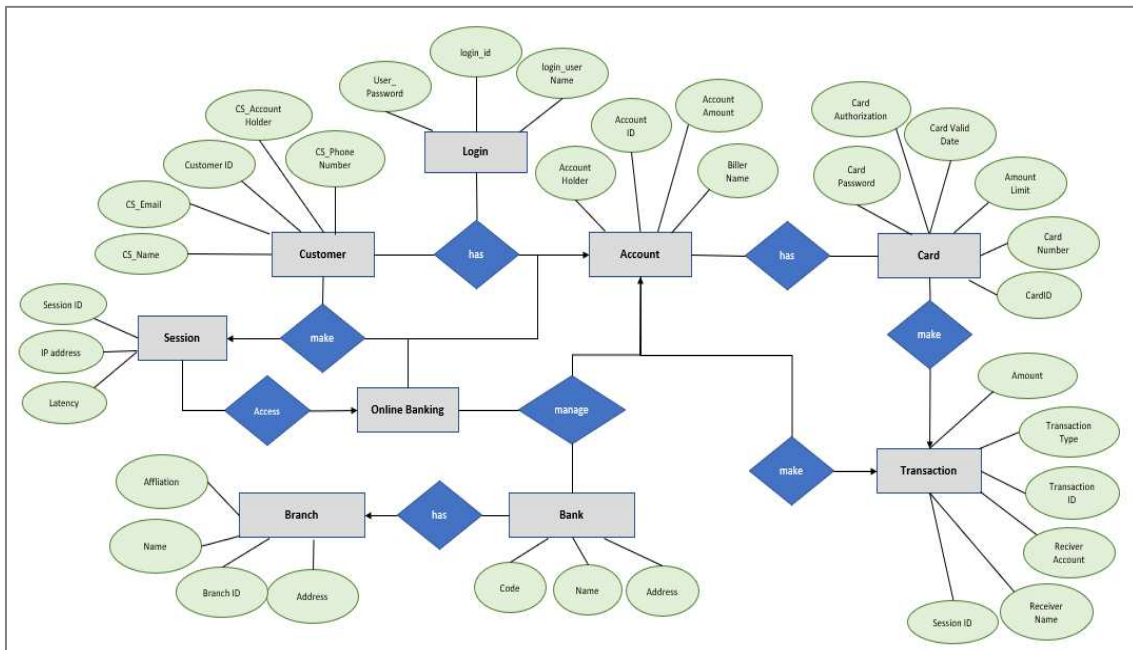


Figure 4-1. ER diagram for an online banking system.

There were originally 120 attributes in the dataset, however, many attributes have over 80% of missing values. Although there were many attributes with a lot of missing values, most of the attributes were static data such as Customer's Surname, Middle Name, First Name, Last Name, Home Address, Office Address, Bank Address, Email 2 and Email 3, Customer's Phone Number 1 and Phone Number 2 and so on. Some attributes are dynamic data related to the web browser and session actions. They are valuable attributes as they represent of part of user's behaviour during the transaction. At least two methods of handling missing values are considered in order to keep them in the dataset. The first method was to fill the missing values with the average numbers or the same content across the same users. However, it was difficult to implement because most of the attributes have blanks under the same user. Another method was to fill the missing values with zero or other numbers. But this method had risks of giving an algorithm the wrong implementation. Filling in with zero or other numbers over 80% of the missing values

may cause incorrect user's behaviour on the web browser and session actions. It is worth experimenting with other important and reliable features rather than using features that may give misleading implementations. This is because the main theme of this research is how to create impactful feature values from given attributes. Priority in this research is given to generating effective features from the remaining features and demonstrating their effects. Therefore, the attributes having over 80% missing values were removed from the dataset. After removing these features, the total number of meaningful features for modelling consequently became 41 features such as transaction amount, event, IP address, device information, online access ID, timestamp, customer ID, email domain, account information, etc. as shown in Table 4-1.



Attribute Name	Description
ED_EVENTTYPETX	Type of event e.g., Customer Login, Make Payment etc
ED_TXNID	Transaction ID
ED_CHANNELIDENTIFIER	A way that customers can interact with a bank. This can be via the telephone, internet banking, branch, mobile.
ED_FINANCIALINSTITUTENM	Financial Institute name
ED_SUBCHANNELNM	Sub-channel name
CUSTD_PARTYID	Customer Party ID
CUSTD_EMAILADDRESSTX	Customer's email address
EVENT	Event of transaction
AUTO_RESPONSE	Auto-response
LOGIN_LATENCY	Latency
SEC_LATENCY	Second Latency
IDVD_LOGINTYPE	Login Type
ACTD_BANKACCTNO	Account's bank account number
ACTD_ACCTYPENM	Account type
ACTD_AVAILABLEBALANCE	Available balance
ACCTLGN_LASTUPDATE	Last login date
ACCTLGN_FARSTUPDATE	Frist login date
TRNSD_BENEFICIARYACCTNO	Beneficiary account number
TRNSD_TRNSAM	Transaction amount
TRNSD_LASTBALANCE	Transaction last balance
TRNSD_TXNREFERENCETX	Transaction reference
IDVD_FASTPASS	First pass code
IDVD_SECPASS	Second pass code
IDVD_LASTPASS	Last pass code
IDVD_IPADDRESSID	IP address
IDVD_CLIENTSCREENRESOID	Client screen resolution
IDVD_USERAGENTTX	User-agent
IDVD_DEVICEID	Device ID
IDVD_INTESESSIONID	Internet session ID
IDVD_SCREENSIZE	Client screen resolution
IDVD_Devicetype	Access device type
IDVD_IPADDRESSID	IP address
IDVD_DEVICEINFO	Device Information
IDVD_TELESESSIONID	Telephone session ID
IDVDATA_TRNSTS	Transactions timestamps
EVENT	Event of transaction
ACCESS_CD	Access Code
Last LATENCY	Latency
LATUPDATE	Latest update timestamps
ACTD_AVAILABLECARD	Card Type
<b>Is Fraud</b>	Fraud flag whether fraud or not

Table 4-1. Description of attributes in the original dataset

## 4.2. Exploratory Data Analysis (EDA)

The exploratory data analysis in this section is intended to investigate trends and patterns in data and summarize the key observations. It provides the fundamental understanding of the data, missing values, and distribution. EDA is very helpful for the data preparation process as a first step towards achieving improved machine learning models. The EDA methods in this section mainly consist of histograms, scatter plot and distributions. Through these methods of EDA, I will explore and define the problem statement on the dataset.

In order to initiate exploring the insights of the data, I used Jupiter notebook and imported the dataset. The dataset contains 4,273 fraudulent transactions out of 130,000 transactions with dataset as shown in Figure 4-2.

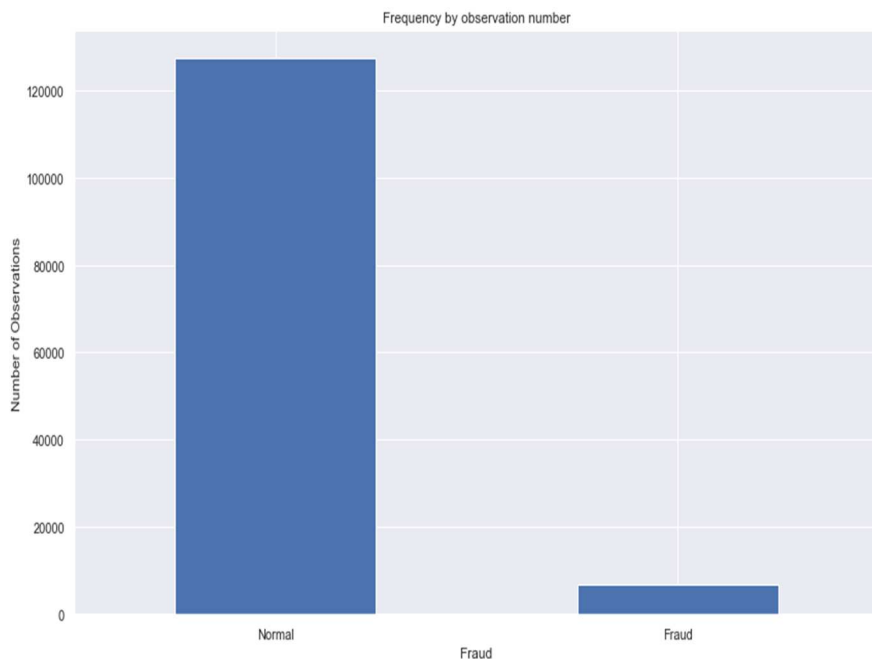


Figure 4-2. Unbalanced dataset

The features used for the EDA in this section include Transaction Amount, the transaction datetime, Auto Response, Access Code, Device Information, Credit Card and Event Type.

The distribution of transaction amount in the dataset is highly skewed to the left as illustrated in Figure 4-3. The fraudulent transaction amount is described in red whereas the legitimate transaction amount is shown in green. In the plot of the distribution of transaction amounts, the mean value of both transactions is between £134.89 and £136.22 while the largest transaction is marked between £3,149 and £5,095 as shown in Figure 4-3.

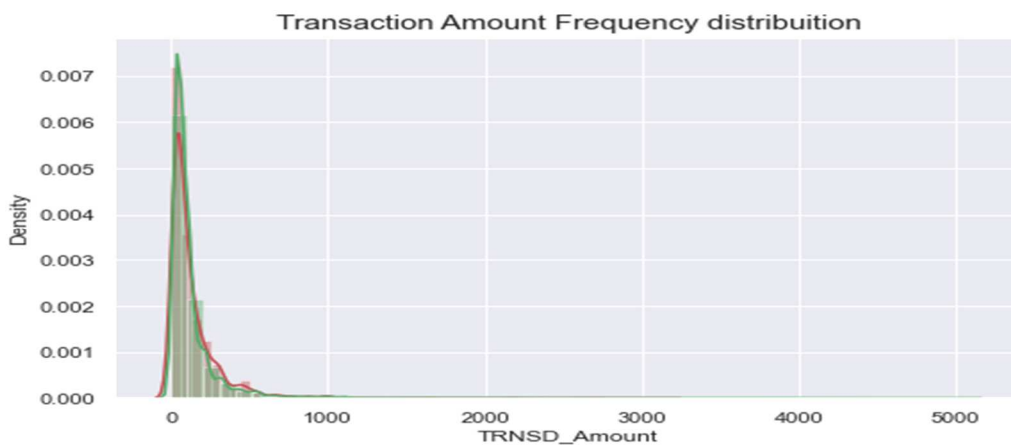


Figure 4-3. The distribution of transaction amount

In Figure 4-4, the boxplot of Transaction Amount by Fraud Flag displays the distribution for each flag, which is a fraud or not. It shows the distribution is different when it reaches to minimum, first quartile, median, third quartile, and maximum.

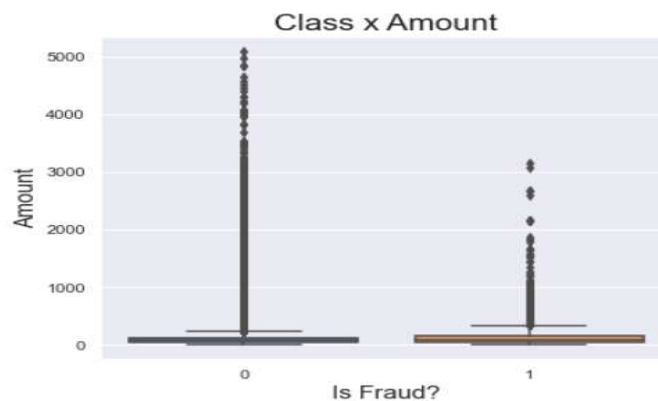


Figure 4-4. Boxplot of transaction amount by fraud frag

In Figure 4-5, the feature of available Balance which indicates the total amount of money the account holder can use is plotted. The Figure 4-5 shows the fraudulent transaction of available balance is in red while the customer's transaction of available balance is in green. The distribution of the available balance is highly skewed to the left as well as the transaction amount.

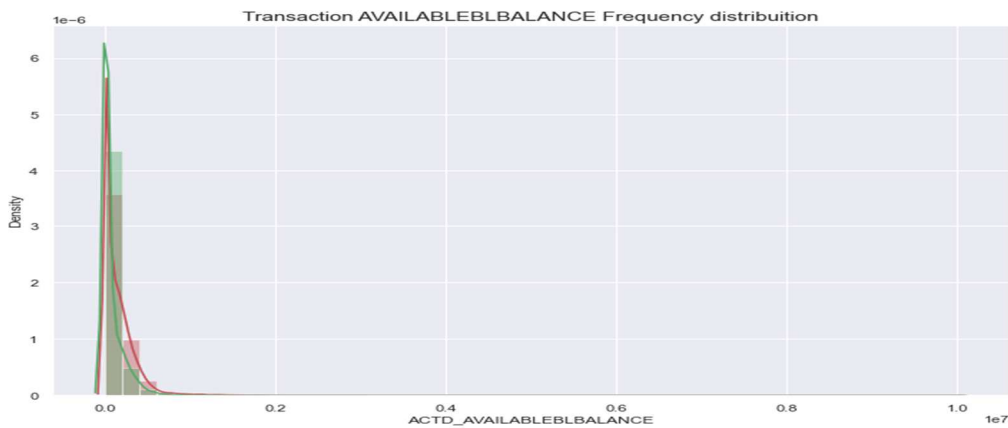


Figure 4-5. Distribution of the transaction frequency of the available balance

Figure 4-6 and figure 4-7 also display the login latency during transactions. It is difficult to discriminate the difference from both tendency of values in fraud and non-fraud.

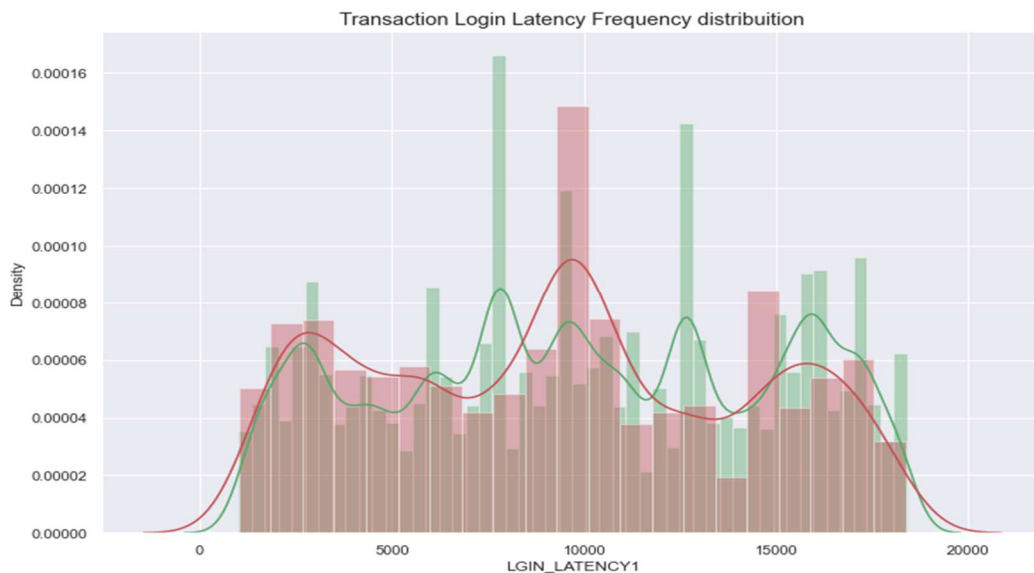


Figure 4-6. Distribution of transaction login latency

Figure 4-7 shows the number of transaction which is flagged as fraud per access device.

The fraudulent transaction has been carried out from both, desktop and mobile.

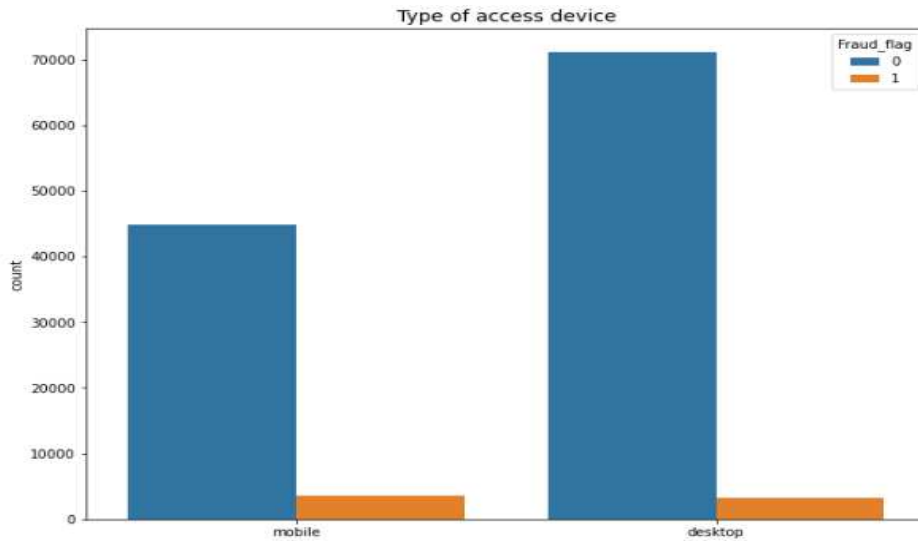


Figure 4-7. Access device type used for transaction

Figure 4-8 displays the number of transaction which is flagged as fraud or not based on the used card type such as American Express, Discover, Visa and Mastercard. The most fraudulent transaction occurred using Visa or Mastercard.

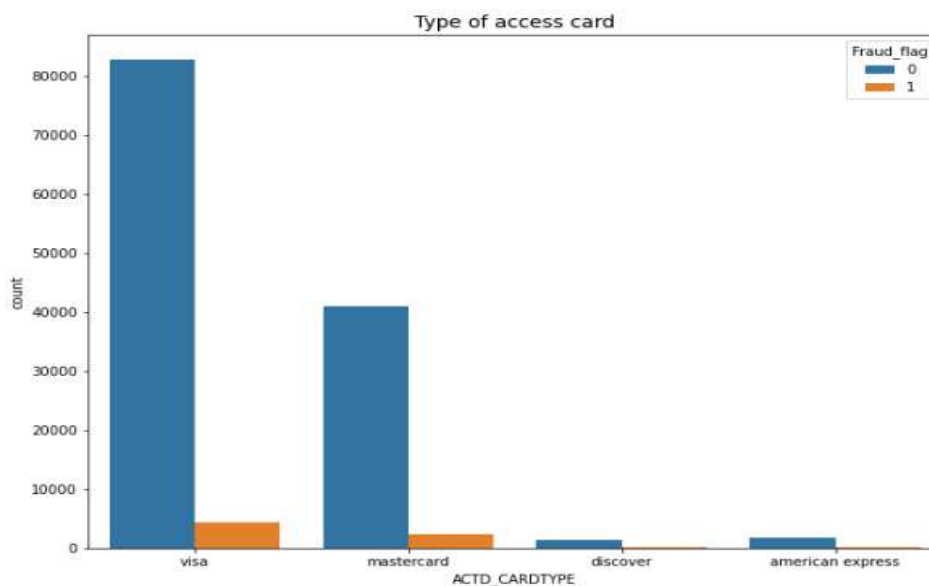


Figure 4-8. Credit card types

Figure 4-9 shows the types of access code which is the key piece of information a user needs to access the internet banking and bank online. The access code in the provided data seems to be only represented by the common initial code. The pattern in orange is frauds and the pattern in blue is normal. At a glance, they are no obvious difference.

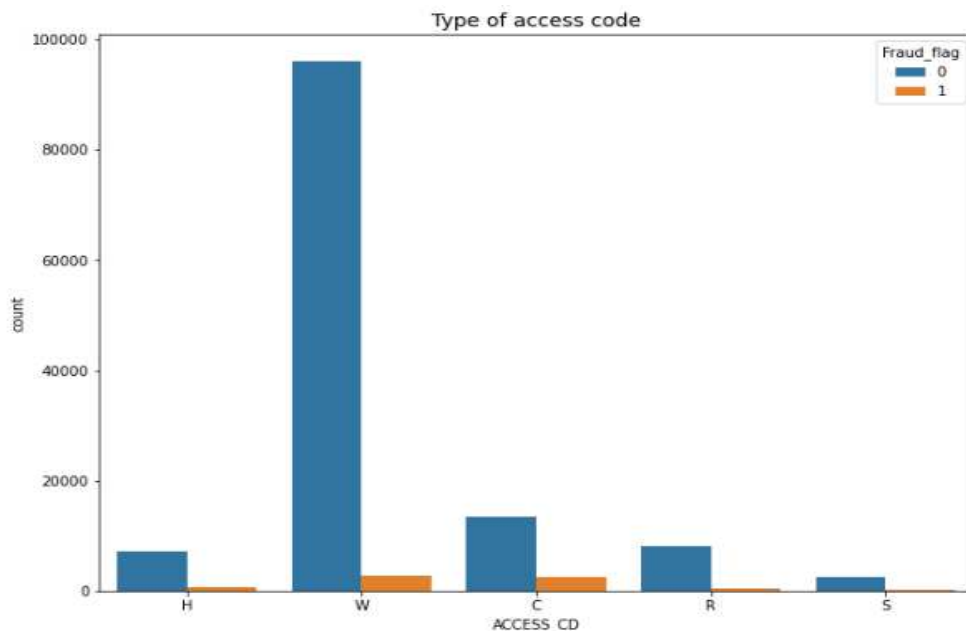


Figure 4-9. Comparison between normal and fraud patterns in access code types

The below figure 4-10 shows various kinds of client's use browsers during the transactions. The blue bars are normal transactions whereas the orange bars are fraudulent ones.

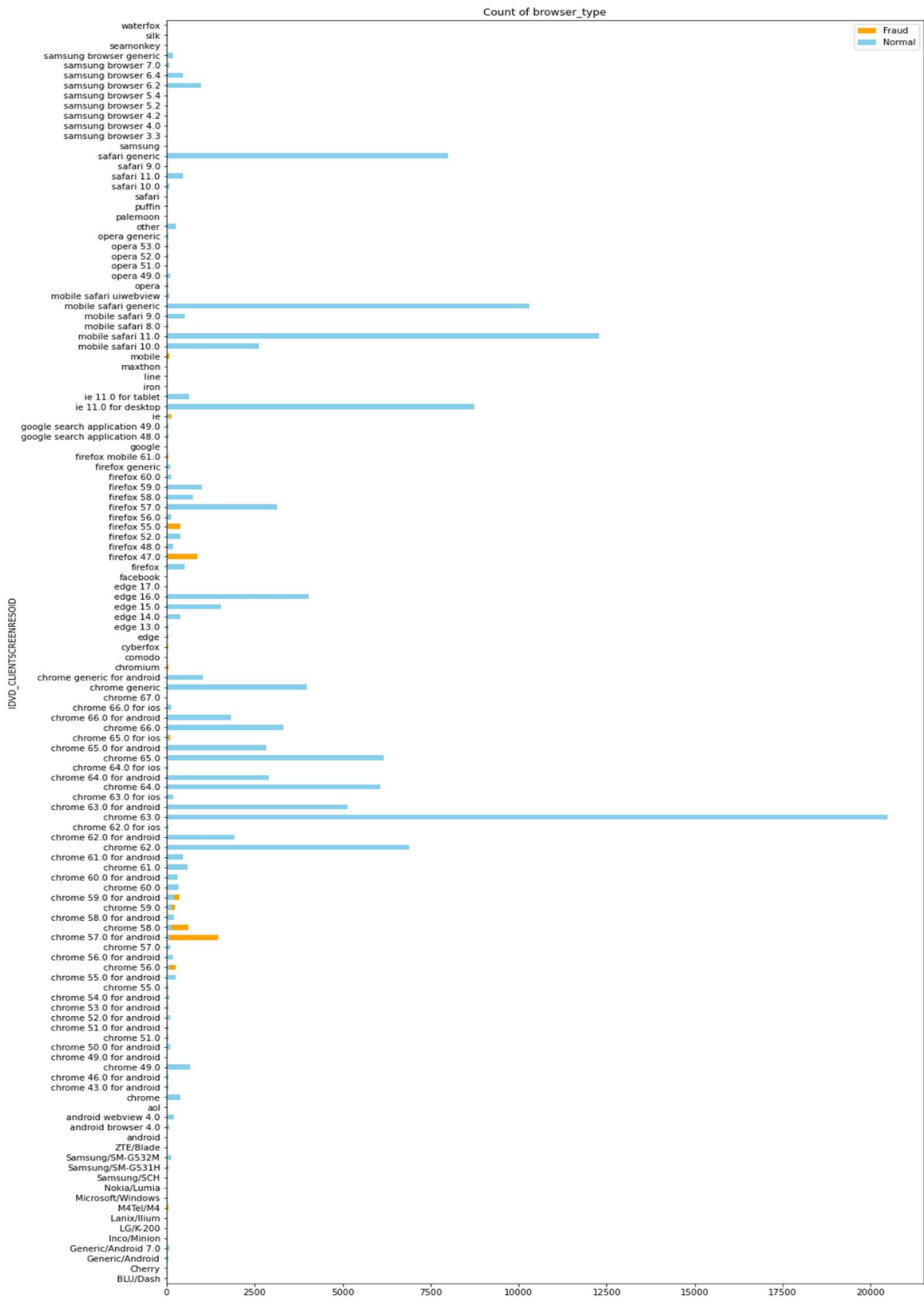


Figure 4-10. Various types of client's screen browser in IDVD\_CLIENTSCREENRESOID

Now, the timestamp is considered as an important feature to discover different behavior between a customer and a fraudster as customers and it will contain their usual lifestyle patterns on a time-series basis. Regarding the number of records in the dataset, the timestamp did not hold a whole sequential time records between November 2015 to June 2016, but it seemed to be fragmentary collected in a certain period. Figure 4-11 shows the number of records between normal transaction and fraud in each month, in which the transaction records were gathered specifically between April and June.

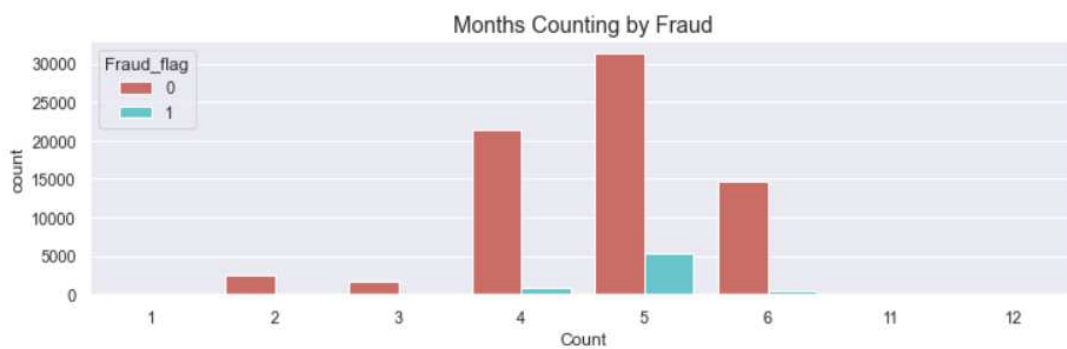


Figure 4-11. Transactions over timestamp in months

Apart from the months, the other related time features such as weekdays, days, and hours were still very significant features that will imply difference between normal and fraudulent behaviour. Figure 4-12 shows all transaction timestamps in the dataset.

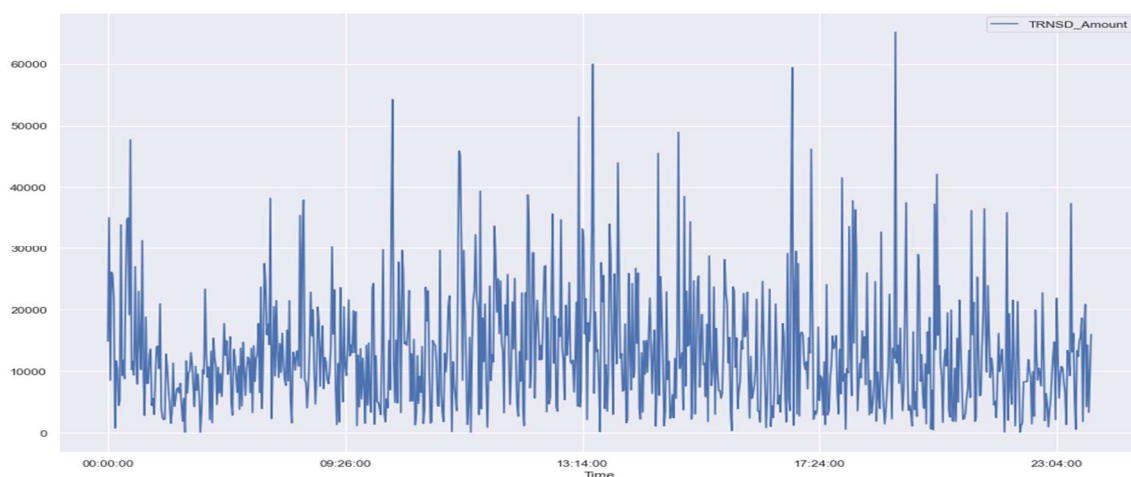


Figure 4-12. Transaction timestamps in the dataset



From this single timestamp feature, I created three different types of time features: Weekdays, Days and Hours, which can be aggregated with other features and produced new features by time slicing in Figures 4-13, 4-14 and 4-15.

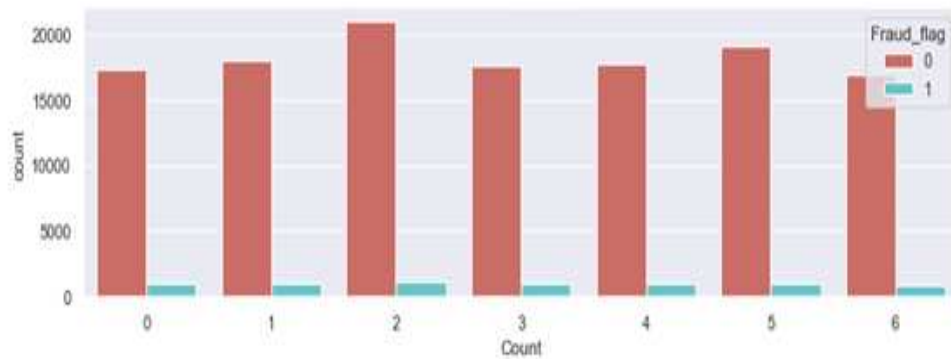


Figure 4-13. Transactions over timestamp in weekdays

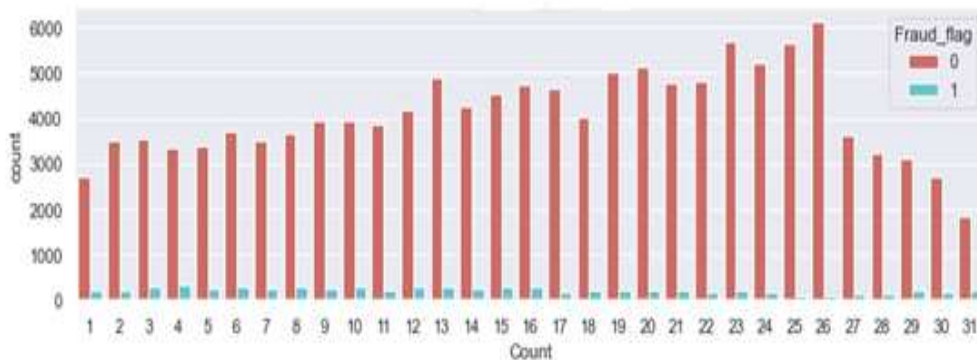


Figure 4-14. Transactions over timestamp in days

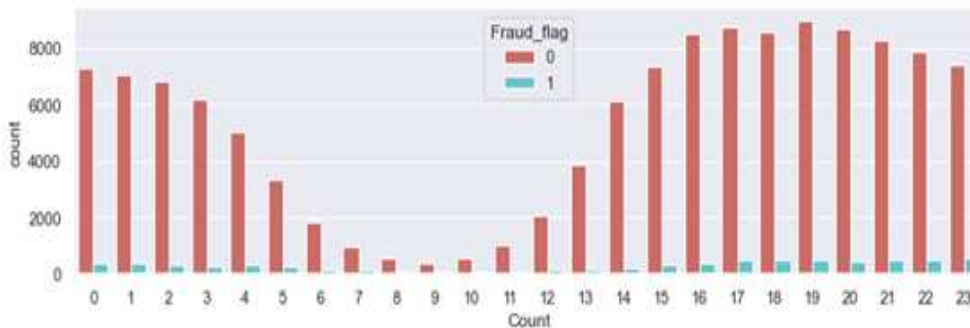


Figure 4-15. Transactions over timestamp in Hours

Specifically, fraud transactions in hours have not occurred from 8 am to 11pm and they seem to become more active after 5pm until midnight.

Throughout the whole visualisations per significant features for fraud detection, the picked-up features in this dataset do not have a remarkable difference between fraud and non-fraud at a glance. In my framework, values in the features will be mapped to the different space to show other aspects of the features by using some techniques of feature engineering. For instance, a logarithm function, which is one of the popular mathematical functions, is applied on Transaction Amount and Available Balance. Tendency of the original values in the feature was shown in Figure 4-3 and Figure 4-5. After applying the log function, the histogram of log transformation is shown in Figure 4-16 and in Figure 4-17. The distribution in green shows normal transactions while the one in red represents fraudulent transactions.

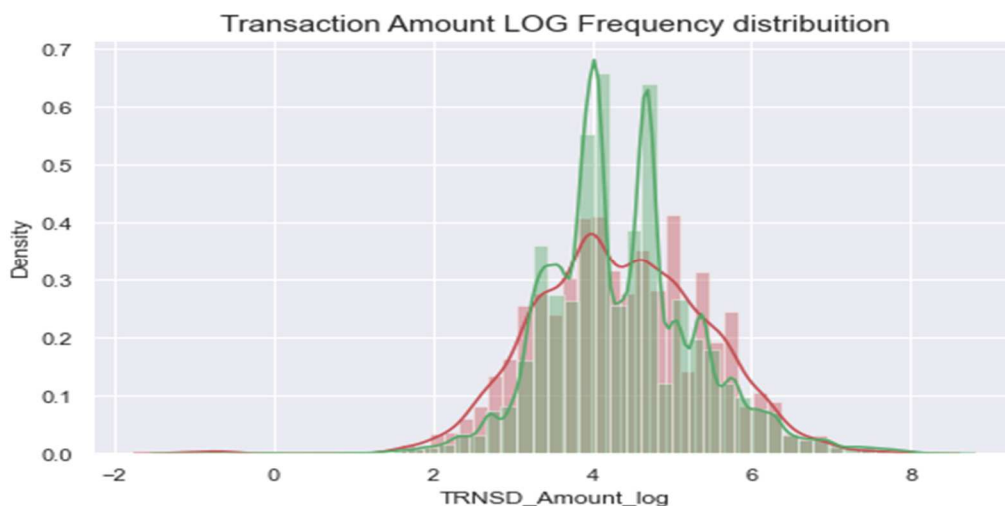


Figure 4-16. Distribution of log transformation amount

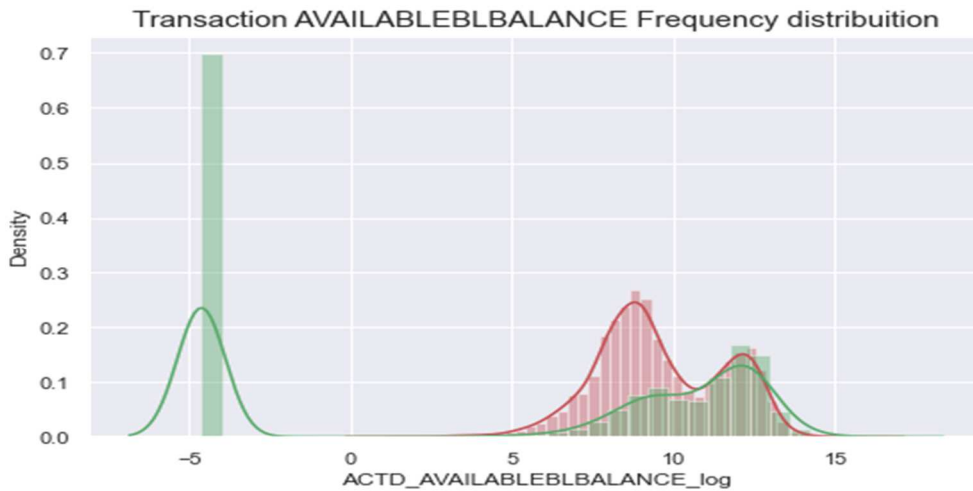


Figure 4-17. Distribution of log transformation available balance

From the visualisation of the log transformation, the distribution graphs provide clearer differences between fraud in red and nonfraud in green than the graphs which are plotted with the original values in Transaction amount and Available Balance. Specifically, the graph in Figure 4-17 shows the left distribution in green can be clearly classified as nonfraud. As another example of visualisation using transformation technique, the transformed feature which applied a function of standard deviation on the login latency of transactions displayed in Figure 4-6 is plotted and shown in Figure 4-18.

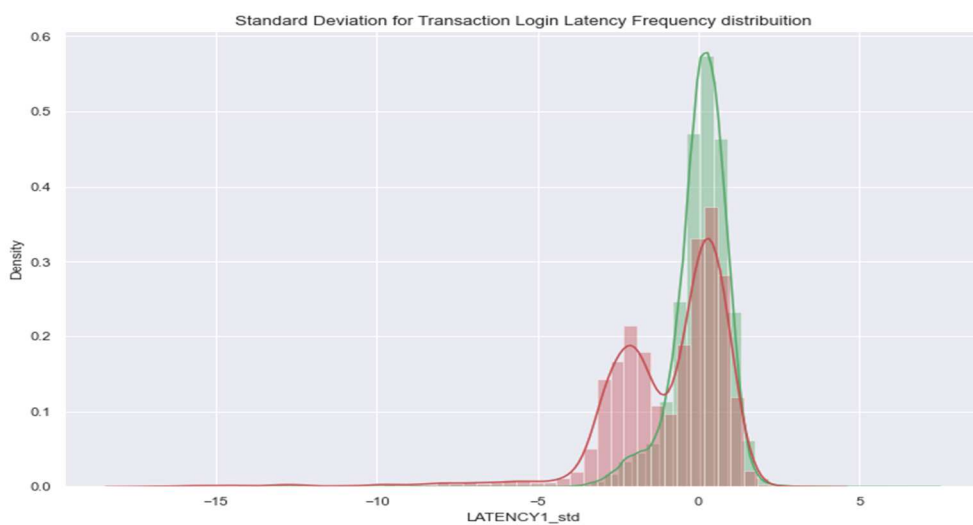


Figure 4-18. Login latency transformed by standard deviation

Figure 4-18 presents the tendency of fraudulent transaction in red. Clearly, a fraudster seems to login faster and carry out the transaction quickly in comparison with the latency of normal transactions shown in green. This tendency could not be seen in the plot of the raw feature shown in Figure 4-6.

### **4.3. Conclusion**

Throughout the outputs in Chapter 4, some key attributes related to customer's behaviour on transaction were visualised and compared with the variables between fraud/non-fraud. Among them, variables in each attribute seemed not to show a clear different tendency between fraud and nonfraud. Therefore, it would be difficult to detect fraudulent transaction by only a rule-based fraud detection system if there is not so much different behaviour on a transaction.

Another insight from the EDA in this section is that the transformed features using techniques of feature engineering could present other aspects of the features which enable a machine learning algorithm to learn the different behaviour of fraud transaction easier. Through the feature engineering framework built in this research, more engineered features can be created and used for machine learning and deep learning which will be addressed in Chapter 5.

## 5. Experiments and Validation of Fraud Detection Framework

### 5.1. Data Preparation processes

As explained details in Chapter 4, a real-life online transaction dataset provided by a European private bank was already integrated from different sources and ready for use as shown in Table 5-1. Thus, the data integration and data modelling processes were not necessary in the experiment. To confirm which attribute needs to be converted from character string to numeric data type, firstly, the data type of each attribute was checked and recognised.

Attribute Name	Data type
CUSTD_PARTYID	int64
ED_EVENTTYPETX	object
ED_TXNID	float64
ED_CHANNELIDENTIFIER	object
ED_FINANCIALINSTITUTENM	object
ED_SUBCHANNELNM	object
CUSTD_EMAILADDRESSTX	object
EVENT	object
AUTO_RESPONSE	float64
LOGIN_LATENCY	float64
SEC_LATENCY	float64
IDVD_LOGINTYPE	int64
ACTD_BANKACCTNO	int64
ACTD_ACCTTYPENM	int64
ACTD_AVAILABLEBALANCE	float64
ACCTLGN_LASTUPDATE	datetime
TRNSD_BENEFICIARYACCTNO	int64
TRNSD_TRNSAM	float64
TRNSD_LASTBALANCE	float64
TRNSD_TXNREFERENCETX	float64
IDVDATA_TRNSTS	datetime
IDVD_FASTPASS	object
IDVD_SECPASS	object

IDVD_LASTPASS	object
IDVD_IPADDRESSID	float64
IDVD_CLIENTSCREENRESOID	float64
IDVD_USERAGENTTX	float64
IDVD_DEVICEID	float64
IDVD_INTESESSIONID	int64
IDVD_SCREENSIZE	float64
IDVD_DeviceType	object
IDVD_DEVICEINFO	object
IDVD_TELESESSIONID	float64
ACCESS_CD	int64
Last_LATENCY	float64
LATUPDATE	datetime
ACTD_AVAILABLECARD	object
Is Fraud	int64

*Table 5-1. Data types of each feature value in the given dataset*

From the above investigation, some features such as available card types, device types, device information, Email domain types, financial institution name, and event types, need to be converted from a string format to a numerical format. Before converting any attributes, as explained in Chapter 3, I need to investigate the contents in each categorical feature and consider which type of encoding methods should apply to the categorical values. Firstly, if the number of categorical types in each attribute are less than ten categories, one-hot encoding is used for converting categorical values into one-hot vectors. If not, I use the label encoding method on the categorical values for transformation to numerical values. Based on the Data Type shown in column in Table 5-1, I plotted the categorical attributes in order as follows.

ACTD_AVAILABLECARD	Counts
Visa	87334
Mastercard	43267
American Express	1936
Discover	1398

*Table 5-2. Category of available cards*

EVENT	Counts
Session Payment Request	122581
Session Request	6278
Password Reset Payment Request	1954
Customer Payment Request	1161
Payment Request	1124
Customer Request	646
Payment Request	495
Product Payment Request	34

Table 5-3. Category of large segment event types

ED_CHANNELIDENTIFIER	Counts
INTERNET_BANKING	103233
TELEPHONY	6808
DIGITAL BRANCH	135

Table 5-4. Category of channel types

Device info	Counts
Desktop	74502
Mobile	48412

Table 5-5. Category of device types

IDVD_FASTPASS	Counts
T	75759
F	48412

Table 5-6. Category of first pass code

IDVD_SECPASS	Counts
T	6782
F	127491

Table 5-7. Category of second pass code

IDVD_LASTPASS	Counts
T	105120
F	29153

Table 5-8. Category of last pass code

ED_FINANCIALINSTITUTENM	Counts
LTB	61548
HAL	40572
BOS	7115
LTS	814
LTJ	55
LTSB	52
HFX	14
LPB	3
LTI	3

Table 5-9. Category of financial institution names

ACCESS_CD	Counts
W	98964
C	16145
R	8671
H	7872
S	2621

Table 5-10. Category of access codes

Although the above nine categorical features could be displayed, the rest of a few features could not be displayed because there were too many types of categories to plot. For instance, the feature value of screen size has 196 types of categories. The feature value of email domain also has 59 kinds of categories. Moreover, the feature value of device information has 1,188 categories. I applied the label encoding method to these feature values for converting numbers. Table 5-11 shows new created features using the one-hot encoding method provided by Python Library, Scikit-Learn OneHot Encoder. The other features are transformed to numerical variables by Scikit-Learn LabelEncoder.

Column	Dtype
EVENT_CustomerRequest	int64
EVENT_PasswordResetPaymentRequest	int64
EVENT_PaymentPaymentRequest	int64
EVENT_PaymentRequest	int64
EVENT_ProductPaymentRequest	int64
EVENT_SessionPaymentRequest	int64
EVENT_SessionRequest	int64



ED_FINANCIALINSTITUTENM_BOS	int64
ED_FINANCIALINSTITUTENM_BoS	int64
ED_FINANCIALINSTITUTENM_HAL	int64
ED_FINANCIALINSTITUTENM_HFX	int64
ED_FINANCIALINSTITUTENM_LPB	int64
ED_FINANCIALINSTITUTENM_LTB	int64
ED_FINANCIALINSTITUTENM_LTI	int64
ED_FINANCIALINSTITUTENM_LTJ	int64
ED_FINANCIALINSTITUTENM_LTS	int64
ED_FINANCIALINSTITUTENM_LTSB	int64
ED_CHANNELIDENTIFIER_DIGITAL BRANCH	int64
ED_CHANNELIDENTIFIER_INTERNET_BANKING	int64
ED_CHANNELIDENTIFIER_TELEPHONY	int64
DeviceType_desktop	int64
DeviceType_mobile	int64
ACTD_AVAILABLEBLCARD_charge card	int64
ACTD_AVAILABLEBLCARD_credit	int64
ACTD_AVAILABLEBLCARD_debit	int64
ACTD_AVAILABLEBLCARD_debit or credit	int64
IDVD_FASTPASS_F	int64
IDVD_FASTPASS_T	int64
IDVD_SECPASS_F	int64
IDVD_SECPASS_T	int64
IDVD_LASTPASS_F	int64
IDVD_LASTPASS_T	int64
ACCESS_CD_C	int64
ACCESS_CD_H	int64
ACCESS_CD_R	int64
ACCESS_CD_S	int64
ACCESS_CD_W	int64
ACTD_CARDTYPE_American Express	int64
ACTD_CARDTYPE_discover	int64
ACTD_CARDTYPE_mastercard	int64
ACTD_CARDTYPE_visa	int64

Table 5-11. New features created by the one-hot encoder method

## **5.2. Feature Creation Processes for Experiment**

Online banking systems in any bank will have multiple common tables and attributes such as transaction amount, time, access information, card information, device information, etc. As I presented the conceptual new framework in Chapter 3, I defined the feature aggregation formula based on a scenario of customer's journey on transaction which uses the six fixed attributes shown in Table 5-12. Following the feature aggregation formula, I carried out the feature creation processes on key feature values linked to customer's transaction behaviour, i.e., Time, Balance, Amount, Latency, Event type, IP address, and Customer ID shown in Table 5-12.

Target Attributes	New created features	Description
Time	LATUPDATE_Weekdays	Last update per weekdays
	LATUPDATE_Hours	Last update per hours
	LATUPDATE_Days	Last update per days
	LATUPDATE_Minute	Last update per minute
Amount	mean_last_Amount	Mean of Last amount
	count_Amount	The count of amount of transactions
	count_last Amount	The count of amount of last transaction
	last_Amount	Last amount of transactinos
	min_last_balance	Maximum last amount
	max_last Amount	Maximum last amount
Balance	mean_last_balance	Mean of balance last transaction
	mean_balance	Mean of Balance
	count_balance	The count of balance of transactions
	last_balance	Last Balance of transactions
	max_balance	Maximum Balance
	min_balance	Minimum last Balance
	mean_latency	Mean of latency
Login / IP address/ Accessing Device	LGIN_LATENCY1	Login with latency1
	LGIN_LATENCY2	Login with latency2
	Last_IPaddress	Last accessed IP address
	Last_login_IPaddress	Last login hours with specific IP address
	Device_IPaddress	Accessed device with IP address
	count_IPaddress	Count of accessed IP address
Event	Event_IPaddress	Specific event with accessed IP address
	Event_device	Specific event with used device
	Event_FINANCIALINSTITUTENM	Specific event with selected financial institution
Customer ID	CUSTD_PARTYID_IDVD_CLIENTSCREENRESOID_ count_LATUPDATE_Weekdays	Count of last update per weekdays by each client's screen resolution based on Customer ID
	CUSTD_PARTYID_IPADDRESSID_count_ LATUPDATE_Weekdays	Count of last update per weekdays by each IP address based on Customer ID
	CUSTD_PARTYID_IDVDATE_TRNSTS_count_ LATUPDATE_Weekdays	Count of last update per weekdays of transaction timestamp by each customer ID
	CUSTD_PARTYID_LAST_LATENCY_count_ LATUPDATE_Weekdays	Count of last update per weekdays by last latency of each customer ID
	CUSTD_PARTYID_TRNSD_TRANSSESSIONCD_ count_LATUPDATE_Weekdays	Count of last update per weekdays by transaction session ID of each customer ID
	CUSTD_PARTYID_ACTD_AVAILBALANCE_ log_count_LATUPDATE_Weekday	Count of last update per available balance of log transformation by each customer ID
	CUSTD_PARTYID_ACCESS_CD_count_ LATUPDATE_Weekdays	Count of last update per access code of each customer ID
	CUSTD_PARTYID_TRNSD_Amount_log_count_ LATUPDATE_weekdays	Count of last update per transaction amount of log transformation by each customer ID
	CUSTD_PARTYID_TRNSD_Amount_count_ LATUPDATE_Weekdays	Count of last update per transaction amount of each customer ID

Table 5-12. New created features by feature aggregation

Then, I applied the five feature transformation methods which were determined to use in the framework as described in Chapter 3 to each attribute related to a transaction. The new transformed features are shown in the below Table 5-13.

Transform Methods	New created features	Description
Confidence Interval Formulas	Balance_min_mean	Minimum mean of Balance
	ACTD_AVAILABLEBALANCE_min_mean	Minimum mean of available balance
	Trans_Amount_min_mean	Minimum mean of Transaction amount
	mean_last_Amount_count	Mean of the count of Last Amount
	min_last Amount	Minimum last amount
	min_last_balance	Maximum last amount
	max_last Amount	Maximum last amount
	max_last_balance	Maximum last Balance
	count_last Amount	The count of amount of last transaction
	count_minimum_balance	The count of total minimum Balane in month
	mean_last Amount	Mean of amount last transaction
	mean_last_balance	Mean of balance last transaction
	max_last_Amount	Maximum amount last transaction
	Log Transformation	Amount_log
ACTD_AVAILABLEBALANCE_log		Log transformation of available balance
Standard Deviation	Balance_min_std	Minimum standard deviation of Balance
	ACTD_AVAILABLEBALANCE_min_std	Minimum standard deviation of available balance
	Trans_Amount_min_std	Minimum standard deviation of transaction amount
	AUTO_RESPONSE_std	Standard deviation of Auto Response
	LATENCY1_std	Standard deviation of latency 1
	LATENCY2_std	Standard deviation of latency 2
	LAST_LATENCY_std	Standard deviation of last latency
	Days_std	Standard deviation of days
	Weekday_std	Standard deviation of weekday
	Hours_std	Standard deviation of hours
K-Means	clusters_1	Cluster 1 by K-Means
	clusters_2	Cluster 2 by K-Means
	clusters_3	Cluster 3 by K-Means
PCA	PCA_EVENT0	PCA of Event
	PCA_EVENT1	PCA of Event
	PCA_FinancialInfo0	PCA of Financial Institution Info
	PCA_FinancialInfo1	PCA of Financial Institution Info
	PCA_FinancialInfo2	PCA of Financial Institution Info
	PCA_CustomerID_IP_Amount	PCA of Customer ID and IP address and Amount

Table 5-13. New created features by feature transformation

### 5.3. Feature Selection Processes for the Experiment

To select appropriate features which will not give a drastic influence on overfitting, there are two processes in the framework as described in Chapter 3: calculating correlation of features and drop some features with very high positive or negative correlation with the other feature, then, measuring feature importance. First, I checked correlation coefficient by using correlation matrix. The number of all attributes in the dataset is too big to plot at a time, thus, I extracted some candidate features to be dropped based on the threshold value above  $\pm 0.9$  as shown in Table 5-14.

X feature	Y feature	Score
ED_SUBSCR_Found	ED_SUBSCR_NotFound	-1
IDVD_FASTPASS_F	IDVD_FASTPASS_T	-1
IDVD_LASTPASS_T	IDVD_LASTPASS_F	-1
IDVD_SECPASS_T	IDVD_SECPASS_F	-1
IDVD_LASTPASS_F	IDVD_LASTPASS_T	-1
LATENCY1_std	PCA_PASS0	-1
ACTD_AVAILABLEBLCARD_debit	ACTD_AVAILABLEBLCARD_credit	-0.99332958
ACTD_CARDTYPE_visa	ACTD_CARDTYPE_mastercard	-0.940520223
AUTO_RESPONSE_std	AUTO_RESPONSE	1
Days_std	LATUPDATE_Days	1
TRNSD_Amount_log	Trans_min_mean	1
Balance_min_mean	Balance_min_std	1
Hours_std	LATUPDATE_Hours	1
ACTD_AVAILABLEBLBALANCE_log	ACTD_AVAILABLEBLBALANCE_min_mean	1
ED_CHANNELIDENTIFIER_INTERNET_BANKING	EIA Code Device_BROWSER	0.987223011
EIA Code Device_BROWSER	ED_CHANNELIDENTIFIER_INTERNET_BANKING	0.987223011
Amount	Amt_to_IDVD_IPADDRESSID	0.957201039
count_balance	count_last	0.903642141
count_last_balance	count_last	0.903642141

Table 5-14. Features with very high correlation coefficient with the other feature.

From the correlation coefficient variables, I extracted features which have very correlation greater than  $\pm 0.90$  and dropped these attributes painted in grey as shown in Table 5-14.

Next, to measure the feature importance, I built a simple random forest model which is an ensemble of decision trees and trained it via the bagging method by using both the original and the created features. I used a library of “RandomForestClassifier” provided by scikit-learn that is an open-source python library. That library is convenient and

optimised for DTs which has all the hyperparameters of bagging classifier to control how trees are grown. Scikit-learn measures feature importance by looking at how much the tree nodes that use that feature reduce impurity on average and computes a score of feature importance automatically for each feature after training. I selected the features which has positive feature importance scores. Which means that I dropped the features with no score of feature importance. The result of feature importance measurement after dropping the irrelevant features is presented in Table 5-15. The feature importance was measured using all features in the dataset including both raw features and new created features.

Features Name	Importance
ED_TXNID	0.146411203
CUSTD_PARTYID_LAST_LATENCY_count_LATUPDATE_Weekdays	0.138431516
IDVD_IPADDRESSID	0.084311684
CUSTD_PARTYID_IDVD_IPADDRESSID_count_LATUPDATE_Weekdays	0.054836879
EIA Code Device_BROWSER	0.050055369
PCA_EVENT0	0.04996359
count_last	0.048155935
count_balance	0.044048671
CUSTD_PARTYID_ACTD_AVAILABLEBLBALANCE_log_count_LATUPDATE_Weekdays	0.0385729
CUSTD_PARTYID	0.036525414
LAST_LATENCY_std	0.034676433
CUSTD_PARTYID_TRNSD_Amount_log_count_LATUPDATE_Weekdays	0.033201245
CUSTD_PARTYID_IDVD_CLIENTSCREENRESOID_count_LATUPDATE_Weekdays	0.031423183
IDVDATA_TRNSTS	0.026768962
LATUPDATE_Days_to_mean_CustomerID	0.019360331
FIRSTLGIN_Days	0.018580164
ACCESS_CD_W	0.015114629
IDVD_SCREENSIZE	0.012545246
Amt_to_mean_ED_TXNID	0.011591451
LATUPDATE_Days	0.009760756
TransactionID	0.008878403
ACCESS_CD_C	0.007942867
TRNSD_BENEFICIARYACCTNO	0.007511596
mean_balance	0.006406679
ED_FINANCIALINSTITUTENM_HAL	0.005652446
max_last_balance	0.005568455
min_last_balance	0.005504727
CUSTD_PARTYID_DeviceInfo_count_LATUPDATE_Weekdays	0.004856424
ED_FINANCIALINSTITUTENM_LTB	0.004042116
IDVD_CLIENTSCREENRESOID	0.003921893
FIRSTLGIN_Hours	0.003887878
LATUPDATE_Weekdays_to_mean_CustomerID	0.003757351
FIRSTLGIN_Weekdays	0.002689806
Amount	0.001679155
FIRSTLGIN_Minute	0.001570854

max_last	0.001544174
TRNSD_TXNREFERENCETX	0.00120954
ACTD_AVAILABLEBLBALANCE_log	0.00113074
ED_SUBSCR_Found	0.001048989
Amt_to_mean_LGIN_LATENCY1	0.001039936
mean_last	0.001007615
max_balance	0.000935558
AUTO_RESPONSE	0.000908164
TRNSD_Amount_log	0.000878105
IDVD_CONF_Found	0.000767246
min_last	0.000671129
LATENCY1_std	0.000646932
EIA Code Device_TELEPHONY	0.00064605
debit_Payment_American_C	0.000641869
Amt_to_mean_CustomerID	0.000562255
LATUPDATE_Weekdays	0.000551129
IDVD_CONF_New	0.000522654
ACCESS_CD_H	0.00052211
ACTD_BANKACCTNO	0.000497427
DeviceInfo	0.000482292
TRNSD_LASTBALANCE	0.000430867
LATUPDATE_Hours	0.000424496
LATENCY2_std	0.000340366
LastACCTLGN_Days	0.000330836
LATUPDATE_Minute	0.00031417
ACTD_AVAILABLEBLBALANCE	0.000281977
BALANCE_log	0.00027607
min_balance	0.000261497
LastACCTLGN_Minute	0.000257389
LATUPDATE_Second	0.00025461
ED_EMAILA	0.000250312
LastACCTLGN_Weekdays	0.000240946
IDVD_IPADDRESSID.1	0.000200519
LastACCTLGN_Hours	0.00015863
credit_Payment_discover_H	0.000138939
IDVD_FASTPASS_F	0.000133314
IDVD_LASTPASS_F	0.000125768
debit_Payment_mastercard_R	0.000120587
credit_Payment_mastercard_R	0.000108201

*Table 5-15. Feature importance measurement*

The above table shows that many features with higher importance rate are the new features created by feature engineering methods in the framework.

Now, I finally had three types of feature sets: (1) the original dataset, (2) the original dataset plus newly created features, (3) the only selected features in accordance with impact of feature importance. In order to evaluate the effectiveness of using feature

created by the feature engineering framework, as described in Chapter 3, I selected machine learning and deep learning algorithms to carry out experiments on these datasets.

## **5.4. Model Preparation**

Now that the data is prepared, let's proceed to build models with the datasets.

### **5.4.1. Split the dataset into Training and Test sets**

Before building the models, as stated in Chapter 2, machine learning algorithms learn from training data to make good performance on feature cases. A dataset was split into two sets: the training set and the test set as seen in Figure 5-1. To observe a machine learning model's behaviour, I train each model with the training set and adjust its parameters during the learning stage. Then I test the machine learning model after the training stage with the test set which is not used in the training set. By measuring how the model performs using the test data, I get an estimate of the generalisation error which tells me how well the model will perform on samples it has never seen before.

To determine the best train-test ratio is not simple. The parameter estimates have a high variance with less training data. On the contrary, less testing data affects high variance in performance measures. The size of dataset indicates a split ratio. When using the same ratio for datasets of different sizes, it is necessary to adjust the sizes of the training and testing sets. This means that if the sizes of the datasets are different, using the same ratio may result in different sizes of the training and testing sets.



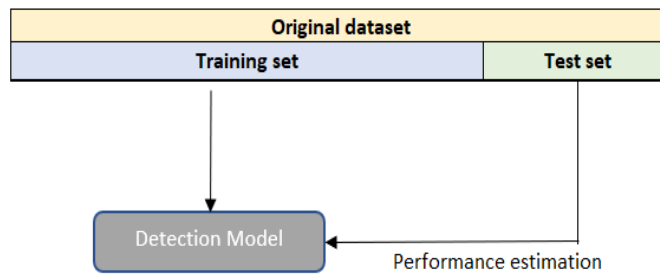


Figure 5-1. Training set and test set

If the size of dataset is smaller than 10,000, the split ratio would be a suitable with 70:30. For more smaller datasets such as less than 1,000, each sample is extremely valuable and cannot be separated any for validation in case of the holdout method. On the other hand, if a very large datasets such as over 1,000,000 is provided, the split ratio can be 99:1 because the size of the test set is still large. In Figure 5-2 shows a way how to consider splitting a dataset [108].

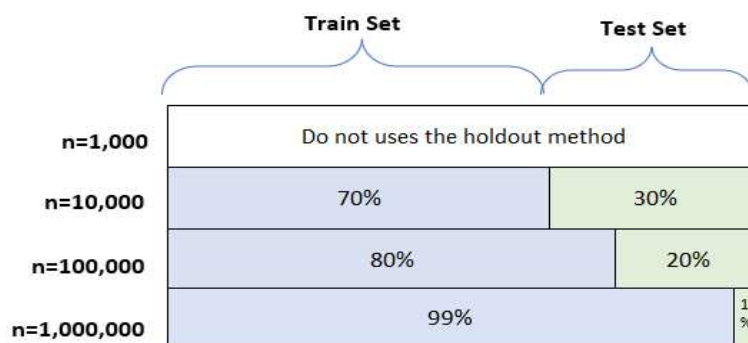


Figure 5-2. A way how to consider splitting a dataset [108]

The dataset in this research contained over 130,000. Therefore, 80:20 was a good starting point as shown in Figure 5-3.



Figure 5-3. The dataset split with 80:20 ratio

While coding in Python, the training data and test data can be created by the following.

```
Train_x, Test_x, Train_y, Test_y =  
    train_test_split(X, Y, test_size=0.2, random_state=RANDOM_SEED)
```

X is all data without a target label. On the contrary, Y has only a target label record without other attributes. The test set size is defined at a rate of 20%.

I ended up having a training set with 107,418 instances (80% of the original dataset) and a test set with 26,855 instances (the remaining 20%). The most significant thing I must consider here is that the ratio of fraud transaction in both training data and testing data should be equal because a prediction model needs to detect fraud under the same environment where the model was built and will be used for. As shown in Table 5-16, both ratio of fraud in training and testing data are about 5%, which is also the same ratio of fraud with the whole dataset.

	# Of Non-Fraud Transaction	#Of Fraud Transaction	Ratio of Fraud
Training data	107,418	5,532	5%
Testing data	26,855	1,392	5%

*Table 5-16. The dataset split into training and testing*

## 5.4.2. Modelling

To evaluate the effective feature sets, I selected five types of algorithms which are Support Vector Machine (SVM), Random Forest (RF), Isolation Forest (IF), Local Outlier Factor (LOF), and Autoencoder (AE) and built fraud detection models with original features and created features. To implement these models, I installed the Anaconda distribution of Python with version 3.6 and created an isolated Python 3.6 environment.

The packages of SVM, RF, IF, and LOF algorithms were provided from scikit-learn. As to AE, I installed TensorFlow and Keras to build the model. TensorFlow is also an open-source program provided by Google. Keras is an open-source library for neural networks that is used on the top of TensorFlow to develop a deep learning model.

### **(A) Support vector machine**

As I described in Chapter 3, there are multiple types of SVM methods; the SVM classifier and the one-class SVM algorithm for commonly used as an anomaly detection model that enables to handle unbalanced classification problems. In the experiment, I selected the one-class SVM algorithm and used a RFB kernel with the default parameter settings instead of manually arranging and adjusting the optimal settings. Conveniently, a sklearn library provides a package of “sklearn.svm.OneClassSVM” for building a model easily. There are some parameters I need to set to in the One-Class SVM algorithm. For instance, degree refers to the degree of polynomial function. Gamma is the kernel coefficient which defines how loosely the model will fit the training data.

```
#One-Class SVM
OneClassSVM (kernel='rbf', degree=3, gamma="auto", max_iter=-1,
random_state=state)
```

### **(B) Random Forest**

A random forest classifier is built using the Scikit-learn library. The RF algorithm is based on ensemble learning and combines multiple decision trees. I use default setup in the library as below.

The RandomForestClassifier class of the sklearn.ensemble library is adapted to solve classification problems via random forest. The n\_estimators parameter is the most

important in the library and refers to the number of trees in the forest. I used the default value of 100 for building the random forest model.

```
#Random Forest Classifier
from sklearn.ensemble import RandomForestClassifier
rfc= RandomForestClassifier(n_estimators=100, random_state=state)
```

### **(C) Isolation Forest**

The sklearn library provides the package of isolation forest algorithm for anomaly detection. In the package, some settings are required. For instance, “n\_estimators” refers to the number of trees in the forest and the default value is 100. “max\_samples” is the number of samples to be drawn to train each base estimator. The default value is “auto”. Contamination refers to the expected proportion of outliers in the dataset. Thus, I set up the outlier fraction as the ratio of fraud in the dataset.

```
# Isolation Forest
from sklearn.ensemble import IsolationForest
IsolationForest (n_estimators=100, max_samples='auto',
contamination=outlier_fraction, random_state=state)
```

### **(D) Local Outlier Factor**

The local outlier factor algorithm computes the local density deviation of a given data point with respect to its neighbours. I imported the package of “sklearn.neighbors.LocalOutlierFactor” for local outlier factor as well as the above algorithms and applied with the default setup of each parameter as below: “contamination” refers to the proportion of outliers in the dataset. “n\_neighbours” is the number of neighbours and is set to 20 as the default value.

```
# local Outlier Factor
from sklearn.neighbors import LocalOutlierFactor
LocalOutlierFactor(n_neighbors=20, contamination="auto")
```

## **(E) Autoencoder for Fraud Detection**

As mentioned in Chapter 2, autoencoder uses only legitimate transactions data during the training and tries to minimize the reconstruction error. The autoencoder model detects the fraudulent transaction by correcting the weight of the reconstruction loss. Therefore, it is difficult to reconstruct the input data when its data is the fraudulent transaction, and the reconstruction error will be higher. Fraud transactions are detected based on points where the reconstruction loss is larger than a fixed threshold.

As mentioned in Chapter 3, I needed to determine some parameters and functions as shown below before I moved into the specific architectures of the autoencoder model:

- Number of hidden layers
- Optimizer
- Loss function
- Number of epochs
- Batch size
- Select number of threshold

In the experiment, it starts with a four-layer autoencoder consisting of a three-hidden layer plus a single output layer. In order to compile the layers for the autoencoder neural networks, I needed to select a loss function which guides the learning of the weights, and

an optimizer to set the process by which the weights are learned. The optimizer I selected is the Adam Optimise. Regarding loss function, I used mean squared error as the evaluation metric for the experiment. Next, I needed to select the number of epochs for training and fitting the model. I carried out this to 1,000 and set the batch size to a generic 128 samples to start with. To use the test set to evaluate how successively this autoencoder can identify fraud in financial transactions dataset I will need to create a test set with twenty percent of the data and labels. In summary, I set up the parameters as shown in Table 5-17.

Parameter Name	Value
Optimiser	Adam Optimise
Loss Function	Mean_Squared_Error
# of Epoc	1000
Batch Size	128
Test_size	0.2

*Table 5-17. Parameters of Autoencoder*

To build the model with the above determined parameters, the library of autoencoder is provided by TensorFlow and Kearas which are available to run on multiple CPUs and GPUs making fast performance. While training the samples, autoencoder encodes and compresses feature values and tries to represent the input data by using seven fully connected layers with 18, 10, 6, and 6 respectively. The first three layers are used for my encoder, the last two go for the decoder. Furthermore, L1 regularization, which is called Lasso Regression, is used during training. Regarding the number of neurons, it depends on the selected dataset. When the model is built with dataset 1, which is the original dataset, the number of neurons becomes 41. Whereas when the model is built with dataset 2 or dataset 3, the number of neurons becomes 100 and 57 respectively.

```

input_layer = Input(shape=(input_dim, ))
encoder = Dense(encoding_dim, activation="tanh",
activity_regularizer=regularizers.l1(learning_rate))(input_layer)
encoder = Dense(hidden_dim1, activation="elu")(encoder)
encoder = Dense(hidden_dim2, activation="tanh")(encoder)
encoder = Dense(hidden_dim3, activation="tanh")(encoder)
encoder = Dense(hidden_dim3, activation="elu")(encoder)
decoder = Dense(hidden_dim2, activation='elu')(encoder)
decoder = Dense(hidden_dim1, activation='tanh')(decoder)
decoder = Dense(input_dim, activation='elu')(decoder)
autoencoder = Model(inputs=input_layer, outputs=decoder)

```

The parameters for training the model are 1,000 epochs with a batch size of 128 samples which is shown in Table 5-17. In order to build an autoencoder model, Keras functional API which is a way to create models with multiple inputs or outputs for Python that is integrated with TensorFlow was utilised.

#### # Autoencoder

```

from keras.models import Model, load_model
from keras.layers import Input, Dense
autoencoder.compile(optimizer='adam',metrics=['accuracy'],loss='mean_squared_error')
nb_epochs = 1000
batch_size = 128
history = autoencoder.fit(x=train_x, y=train_x, epochs=nb_epoch, batch_size=batch_size,
shuffle=True, validation_data=(test_x, test_x), verbose=1, callbacks=[cp, tb]).history

```

## 5.5. Model Preparation

The effectiveness of the feature engineering framework is measured through a comparison with the performance of the building models with the three different types of datasets. As stated in the previous section, to evaluate the efficiency of the created and selected features, the model performance is assessed by AUC, recall, precision, and F-measure.

Here is the information of the three datasets which were used to build each model.

- Dataset 1: only raw features in the original dataset.
- Dataset 2: raw features and all new features.
- Dataset 3: only selected features

### (A) Support Vector Machine model

Table 5-18 displays performance of the one-class SVM model built with dataset 1 as a baseline model. In that Table, there are two results of three-evaluation metrics: precision, recall, and f1-score, for the 0-class case as normal transactions and the 1-class case as fraudulent transactions respectively. In this case, I focus on each result of the metrics in the 1-class case which shows how many detected fraudulent transactions correctly.



SVM classifier (Baseline) model				
	precision	recall	f1-score	
0	0.75	0.16	0.26	
1	0.00	0.03	0.00	
accuracy			0.15	
macro avg	0.38	0.09	0.13	
weighted avg	0.71	0.15	0.25	

Table 5-18. The results of each evaluation metrics of one-class SVM model with dataset 1

As shown in Figure 5-4, the AUC 0.09 was very bad and indicates that the model could not classify the target correctly.

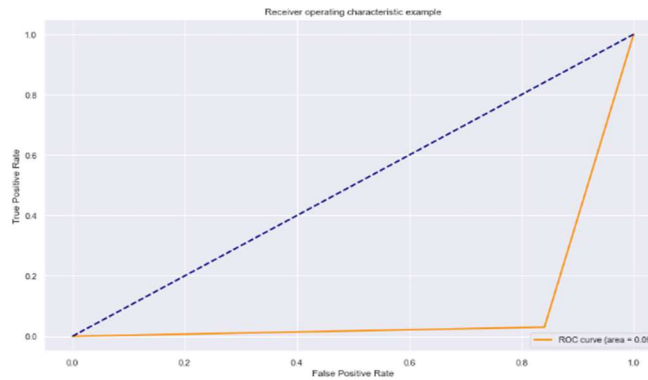


Figure 5-4. AUC of the one-class SVM model built with dataset 1

Table 5-19 displays performance of the one-class SVM model built with dataset 2. Each result got better scores than the first model built with dataset 1. Specifically, the score of recall significantly increased.

Classification Report (All Features) model:				
	precision	recall	f1-score	
0	0.98	0.43	0.6	
1	0.08	0.87	0.14	
accuracy			0.45	
macro avg	0.53	0.65	0.37	
weighted avg	0.94	0.45	0.57	

Table 5-19. The results of the One-Class SVM model with dataset 2 in the three-evaluation metrics

The AUC score of the model became 0.65 as shown in Figure 5-5. It was surprisingly improved than the AUC 0.09 of the baseline model.

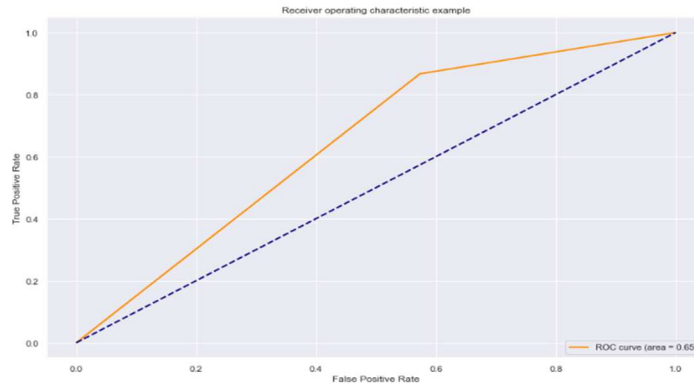


Figure 5-5. AUC of the one-class SVM model built with dataset 2

Lastly, performance of the one-class SVM model built with dataset 3 are shown in Table 5-20. All performance became much better than the first SVM model. Comparing with performance of the model built with dataset 2, the recall score of the model built with dataset 3 become lower. However, other evaluation scores became better than the model built with dataset 2. Furthermore, the prediction ratio of the 0-class case also became much better than the other models as shown in Table 5-20.

Classification Report (Selected Features) mdoel:				
	precision	recall	f1-score	
0	0.96	0.94	0.95	
1	0.19	0.28	0.23	
accuracy				0.9
macro avg	0.57	0.61	0.59	
weighted avg	0.92	0.9	0.91	

Table 5-20. The results of the One-Class SVM model with dataset 3 in the three-evaluation metrics

The AUC score of the model built with dataset 3 was 0.61 as illustrated in Figure 5-6.

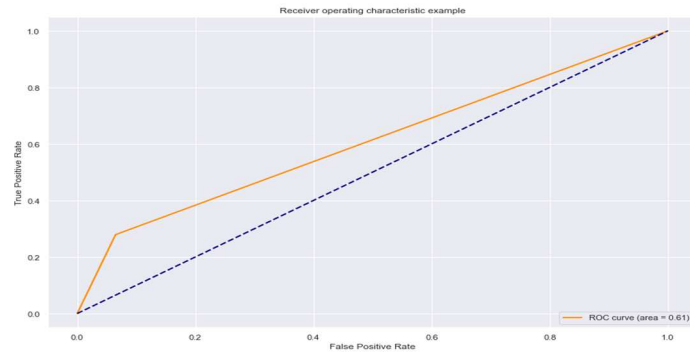


Figure 5-6. AUC of the one-class SVM model built with dataset 3

## (B) Random Forest Models

The random forest algorithm was used to build three RF models using the different types of datasets: dataset 1, dataset 2, and dataset 3. Performance of the first RF model built with dataset 1 are shown in Table 5-21.

Classification Report (Baseline) model:				
	precision	recall	f1-score	
0	0.97	1	0.99	
1	1	0.48	0.64	
accuracy			0.97	
macro avg	0.99	0.74	0.82	
weighted avg	0.97	0.97	0.97	

Table 5-21. The results of the RF model with dataset 1 in the three-evaluation metrics

The baseline RF model became already good performance in each evaluation metric.

The AUC score also became 0.74 as seen in Figure 5-7.

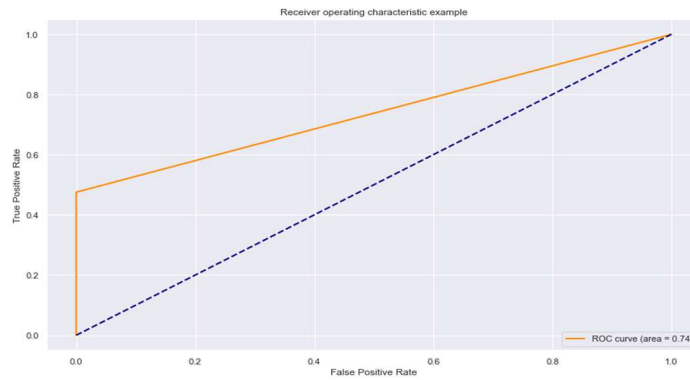


Figure 5-7. AUC of the RF model built with dataset 1

Next, the random forest algorithm built with dataset 2. The results show in Table 5-22.

Classification Report (All Features) model:				
	precision	recall	f1-score	
0	0.99	0.99	0.99	
1	0.9	0.88	0.89	
accuracy				0.99
macro avg	0.94	0.94	0.94	
weighted avg	0.99	0.99	0.99	

Table 5-22. The results of the RF model with dataset 2 in the three-evaluation metrics

The AUC score also became 0.93 as shown in Figure 5-8.

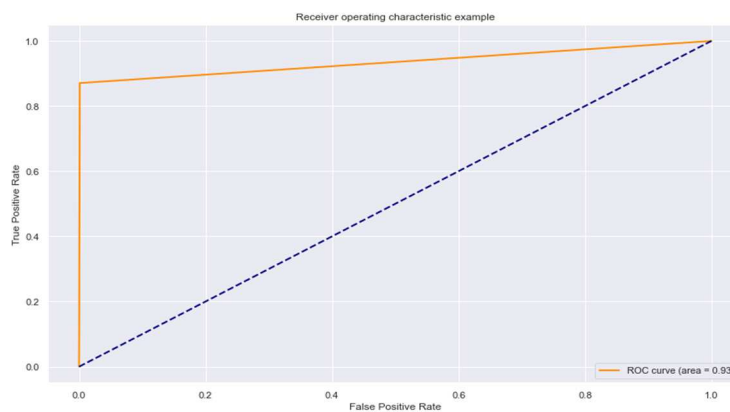


Figure 5-8. AUC of the RF model built with dataset 2

Lastly, performance of the RF model built with dataset 3 are shown in Table 5-23.

Classification Report (Selected Features) model:			
	precision	recall	f1-score
0	0.99	1	1
1	0.99	0.88	0.93
accuracy			0.99
macro avg	0.99	0.94	0.96
weighted avg	0.99	0.99	0.99

Table 5-23. The results of the RF model with dataset 3 in the three-evaluation metrics

The AUC score was 0.94 (see Figure 5-9) and became slightly better than the model built with dataset 2.

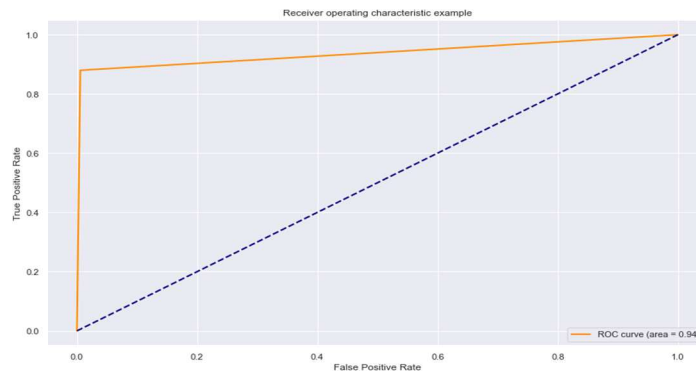


Figure 5-9. AUC of the RF model built with dataset 3

### (C) Isolation Forest Models

In order to build the IF models built with each dataset, as stated in Chapter 3, I used the scikit-learn python library and obtained the following results.

First, Table 5-24 shows performance of the IF model built with dataset 1.

Isolation Forest (baseline) model			
	precision	recall	f1-score
0	0.95	0.96	0.95
1	0.19	0.16	0.17
accuracy			0.91
macro avg	0.57	0.56	0.56
weighted avg	0.90	0.91	0.90

Table 5-24. The results of the IF model with dataset 1 in the three-evaluation metrics

The prediction results in the 0-class case were higher than the prediction results in the 1-class case. The AUC score was 0.55 as displayed in Figure 5-10.

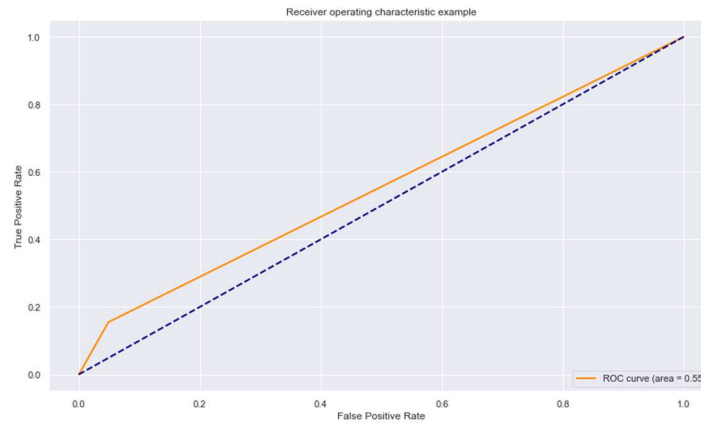


Figure 5-10. AUC of the IF baseline model

Next, performance of the model built with dataset 2 describes in Table 5-25.

Isolation Forest (All Features) model				
	precision	recall	f1-score	
0	0.96	0.96	0.96	
1	0.30	0.31	0.30	
accuracy			0.93	
macro avg	0.63	0.64	0.63	
weighted avg	0.93	0.93	0.93	

Table 5-25. The results of the IF model with dataset 2 in the three-evaluation metrics

The AUC score was 0.64 as seen in Figure 5-11.

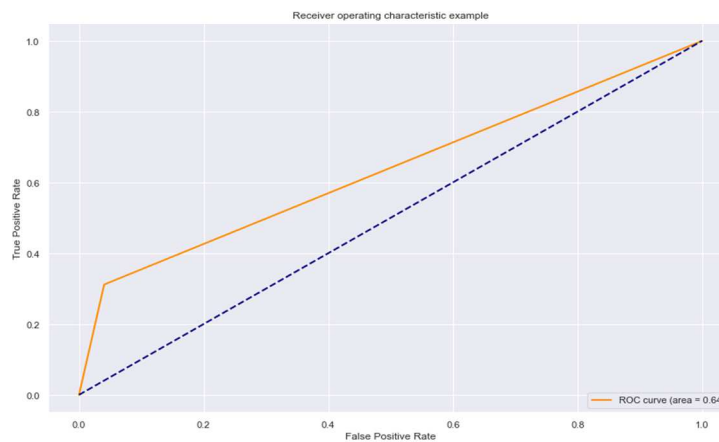


Figure 5-11. AUC of the IF model built with dataset 2

Lastly, performance of the IF models with dataset 3 became as presented in Table 5-26.

The scores of all metrics in the 1-class case doubled the scores of IF baseline model.

Isolation Forest (Selected Features) model				
	precision	recall	f1-score	
0	0.97	0.96	0.96	
1	0.34	0.36	0.35	
accuracy			0.93	
macro avg	0.65	0.66	0.66	
weighted avg	0.93	0.93	0.93	

Table 5-26. The results of the IF model with dataset 3 in the three-evaluation metrics

The AUC score was 0.66 as shown in Figure 5-12.

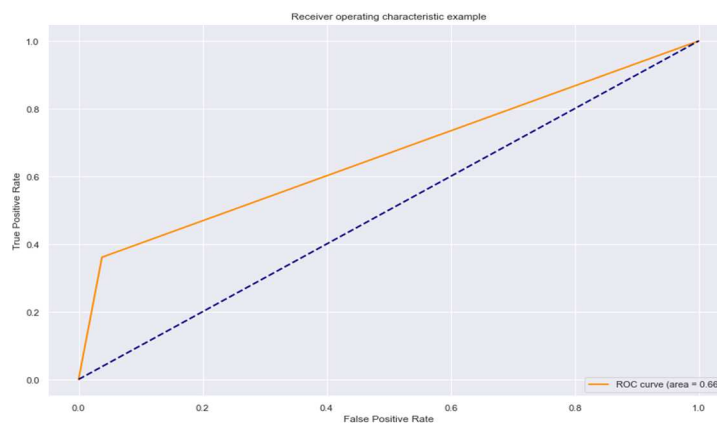


Figure 5-12. AUC of the IF model built with dataset 3

## (D) Local Outlier Factor Models

First, the results of each metrics of local outlier factor model built with dataset 1 displayed in Table 5-27.

Local Outlier Factor (Baseline) model				
	precision	recall	f1-score	
0	0.97	0.95	0.96	
1	0.02	0.04	0.03	
accuracy			0.92	
macro avg	0.50	0.49	0.49	
weighted avg	0.94	0.92	0.93	

Table 5-27. The results of the LOF model with dataset 1 in the three-evaluation metrics

The AUC score became 0.48 in Figure 5-13.

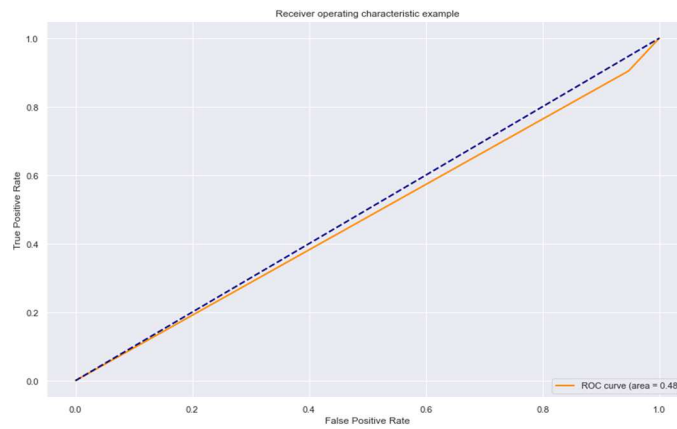


Figure 5-13. AUC of the LOF model built with dataset 1

Next, performance of the LOF model built with dataset 2 is described in Table 5-28.

Local Outlier Factor (All features) model				
	precision	recall	f1-score	
0	0.95	0.95	0.95	
1	0.05	0.05	0.05	
accuracy			0.90	
macro avg	0.50	0.50	0.50	
weighted avg	0.90	0.90	0.90	

Table 5-28. The results of the LOF model with dataset 2 in the three-evaluation metrics

The AUC score of the LOF model was 0.5 as seen in Figure 5-14.

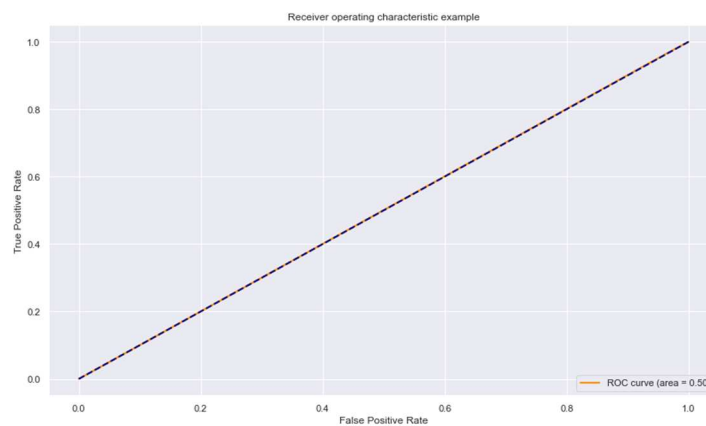


Figure 5-14. AUC of the LOF model built with dataset 2



Lastly, performance of the model built with dataset 3 became in Table 5-29. Comparing with the performances of both the baseline model and the model built with dataset 2, performance of the model built with dataset 3 was improved.

Local Outlier Factor (Selected Features) model			
	precision	recall	f1-score
0	0.95	0.95	0.95
1	0.15	0.15	0.15
accuracy			0.91
macro avg	0.55	0.55	0.55
weighted avg	0.91	0.91	0.91

Table 5-29. The results of the LOF model with dataset 3 in the three-evaluation metrics

The AUC score of the LOF model was 0.55 as illustrated in Figure 5-15.

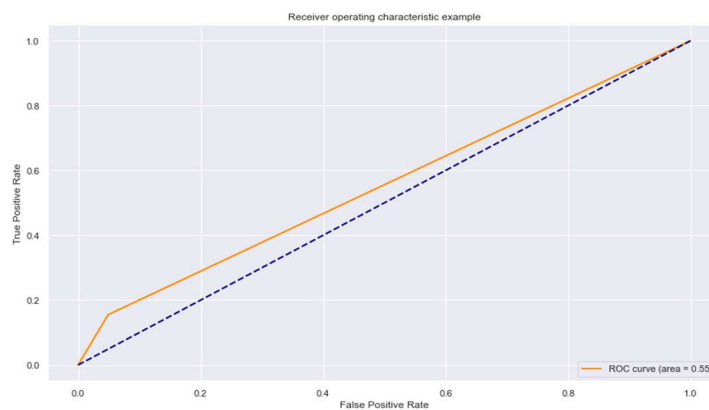


Figure 5-15. AUC of the LOF model built with dataset 3

## (E) Autoencoder models

Building the autoencoder model is not same way with other algorithms. Autoencoders tries to minimise the reconstruction error as part of the training. While training the autoencoder model, I needed to adjust the reconstruction error threshold which I described in Chapter 3. The given threshold is used for determining to be normal or fraud respectively. First, I built the AE models with the reconstruction error threshold 4 and assessed how successively these AE models can identify fraud. Figure 5-16

illustrates the reconstruction error threshold of 4 in dataset 1. The dots above the threshold line demonstrate the true positive and false positive prediction cases. The orange dots show fraudulent transaction and blue dots show normal transaction. In case of a threshold of 4, many orange dots appear under the threshold line.

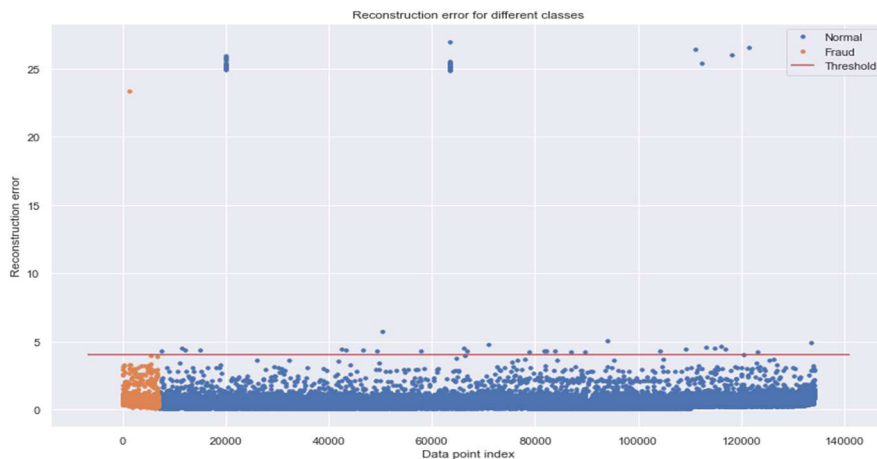


Figure 5-16. The reconstruction error threshold 4 in dataset 1

The aim of determining the threshold is to classify between the orange and blue dots. Therefore, Figure 5-16 illustrates that the threshold 4 is not appropriate line to classify the given data into two classes. Figure 5-17 shows the confusion matrix of the AE model built with dataset 1.

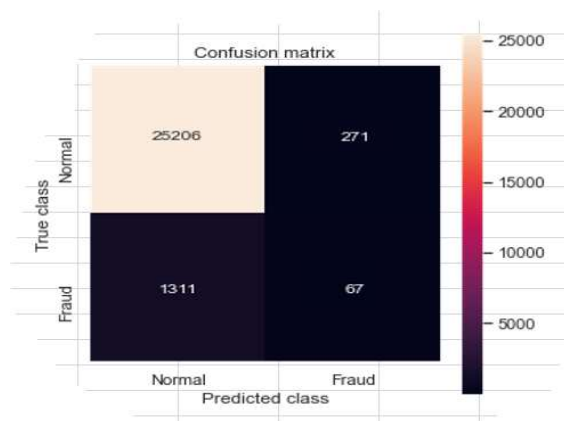


Figure 5-17. Confusion matrix of the autoencoder model with dataset 1 using a threshold of 4

The confusion matrix shows that the number of detected actual frauds is low. The AUC score was 0.65 as shown in Figure 5-18.

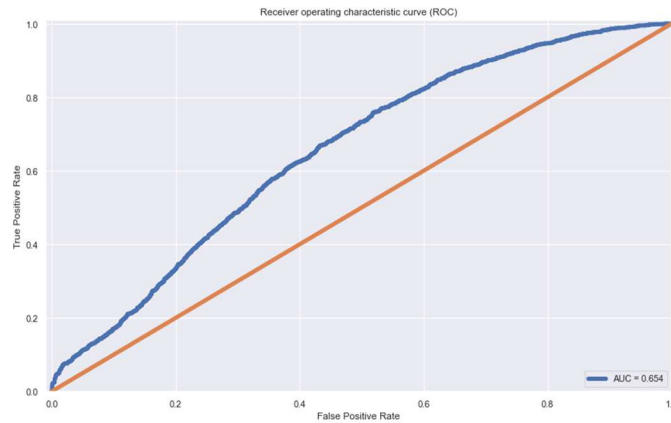


Figure 5-18. AUC of the AE model of threshold4 with dataset 1

The scores of the three-evaluation metrics calculated based on the results of confusion matrix became below in Table 5-30.

Threshold 4	Precision	Recall	F1-score
Dataset 1	0.2	0.05	0.08

Table 5-30. The results of evaluation metrics

Next, Figure 5-19 illustrates the reconstruction error threshold of 4 in dataset 2.



Figure 5-19. The reconstruction error threshold 4 in dataset 2

Newly created features have been added, so several orange dots expand beyond the reconstruction error threshold of 4.

Figure 5-20 shows the confusion matrix of the AE model built with dataset 2.

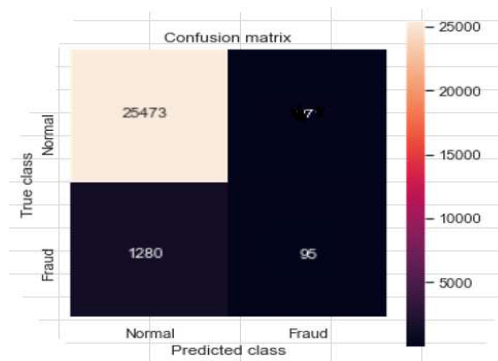


Figure 5-20. Confusion matrix of the autoencoder model with dataset 2 using a threshold of 4

The confusion matrix shows that the number of detected actual frauds become better than the first AE model, but still low. The AUC score was 0.83 as illustrated in Figure 5-21.

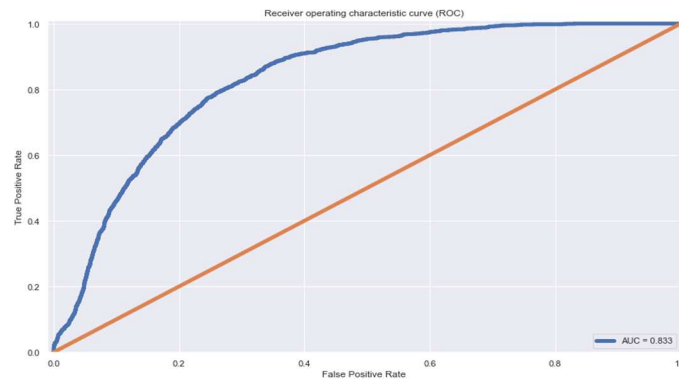


Figure 5-21. AUC of the AE model of threshold 4 built with dataset 2

The scores of the three-evaluation metrics are presented in Table 5-31.

Threshold 4	Precision	Recall	F1-score
Dataset 2	0.93	0.07	0.13

Table 5-31. The results of evaluation metrics

Lastly, Figure 5-22 illustrates the reconstruction error threshold of 4 in dataset 3. The orange dots spread further towards the threshold, making the algorithm easier to separate fraud and the nonfraud.

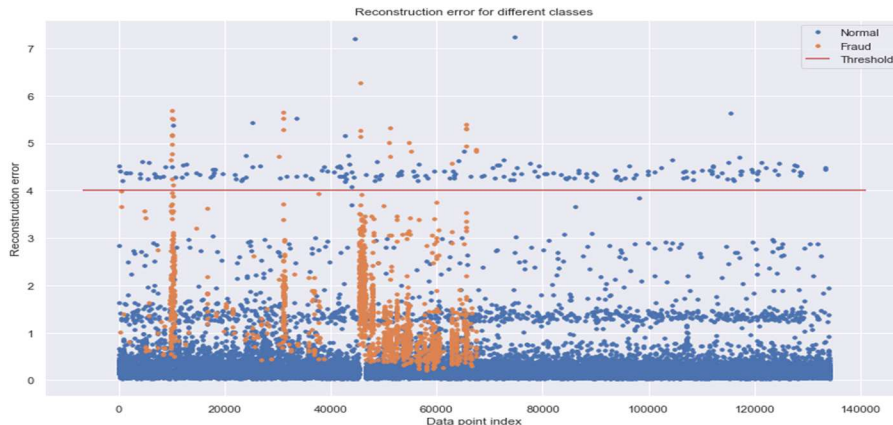


Figure 5-22. The reconstruction error threshold 4 in dataset 3

Figure 5-23 shows the confusion matrix of the AE model built with dataset 3.

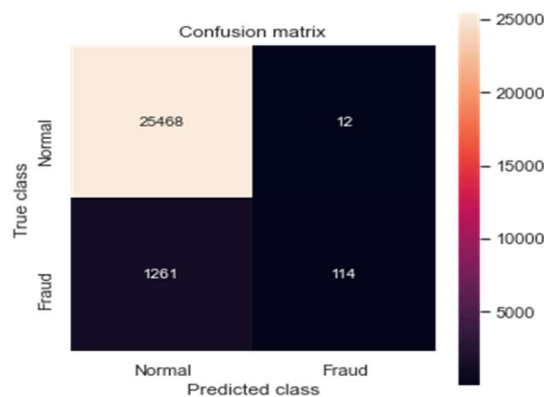


Figure 5-23. Confusion matrix of the autoencoder model with dataset 3 using a threshold of 4

The confusion matrix shows that the number of detected actual frauds become better than the AE model built with dataset 1, but still low. The AUC score became 0.92 as displayed in Figure 5-24.

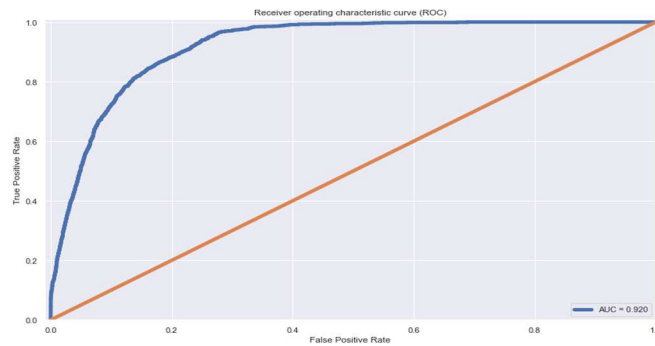


Figure 5-24. AUC of the AE model of threshold 4 built with dataset 3

The scores of the three-evaluation metrics are presented in Table 5-32.

Threshold 4	Precision	Recall	F1-score
Dataset 3	0.9	0.08	0.15

Table 5-32. The results of evaluation metrics

From the above all performance evaluations of the AE models built with each dataset, the reconstruction error threshold 4 was not enough to segment two class dots of orange and blue. Now, I changed the threshold value from 4 to 1 and built the AE models with each dataset.

Figure 5-25 illustrates the reconstruction error threshold of 1 in Dataset 1. By changing the threshold definition from 4 to 1, the accuracy of fraud detection increased, however the number of cases where nonfraud was detected as fraud increased in Figure 5-26.

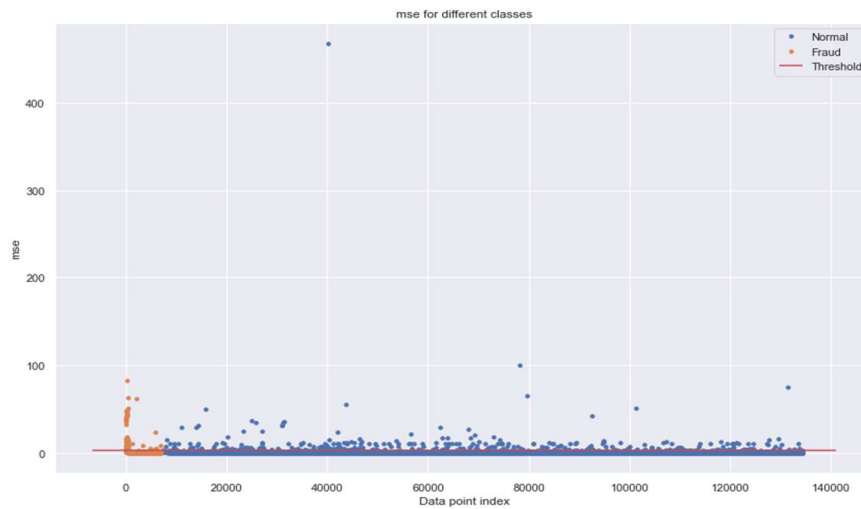


Figure 5-25. The reconstruction error threshold 1 in dataset 1

Figure 5-26 shows the confusion matrix of the AE model built with dataset 1 using the threshold of 1.

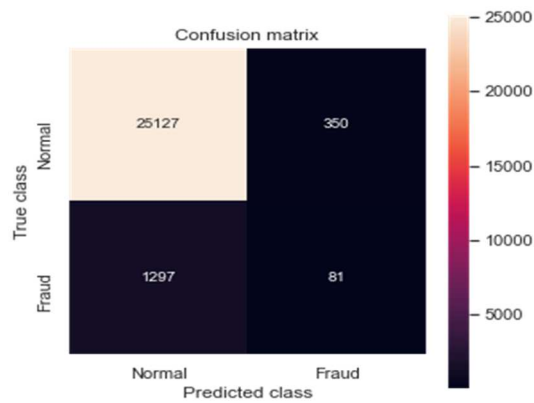


Figure 5-26. Confusion matrix of the autoencoder model with dataset 1 using a threshold of 1

The AUC score became 0.77 and was more than the AE baseline model with the threshold 4 as seen in Figure 5-27.

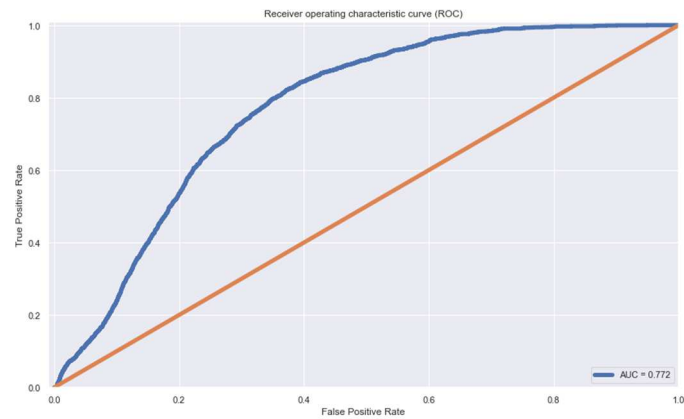


Figure 5-27. AUC of the AE model of threshold 1 built with dataset 1

The scores of the three-evaluation metrics of the AE model built with dataset 1 is displayed in Table 5-33.

Threshold 1	Precision	Recall	F1-score
Dataset 1	0.19	0.06	0.09

Table 5-33. The results of evaluation metrics

Next, Figure 5-28 illustrates the reconstruction error threshold 1 in dataset 2.

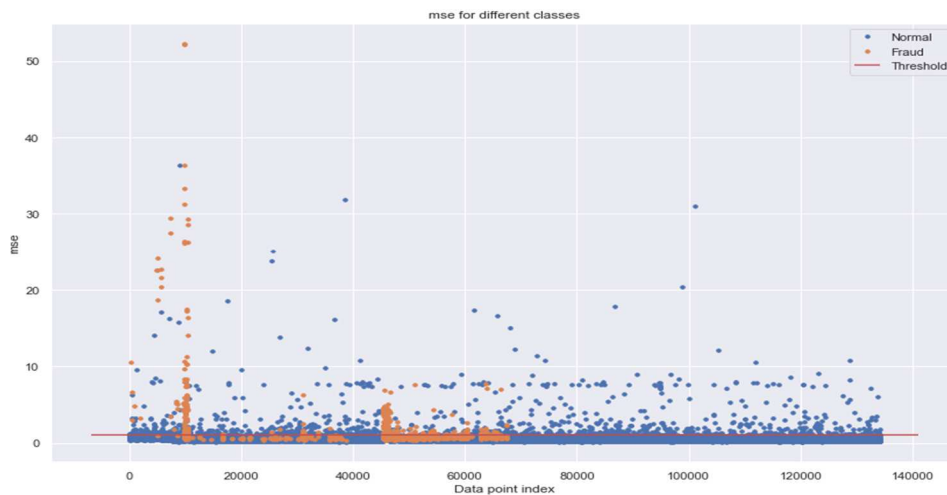


Figure 5-28. The reconstruction error threshold 1 in dataset 2

By shifting the threshold from 4 to 1, the accuracy of fraud detection increased while the number of cases where nonfraud was detected as fraud increased. As a result of lowering the threshold to discriminate more fraudulent data, the accuracy of fraud detection



increased, but it led to the result that non-fraudulent data was identified as fraud. Figure 5-29 shows the confusion matrix of the AE model built with dataset 2 using the threshold of 1.

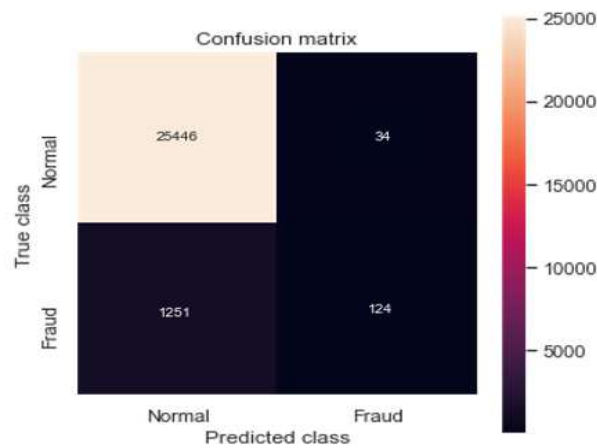


Figure 5-29. Confusion matrix of the autoencoder model with dataset 2 using a threshold of 1

The AUC score became 0.91 and was more than the AE baseline model with the threshold 1 as seen in Figure 5-30.

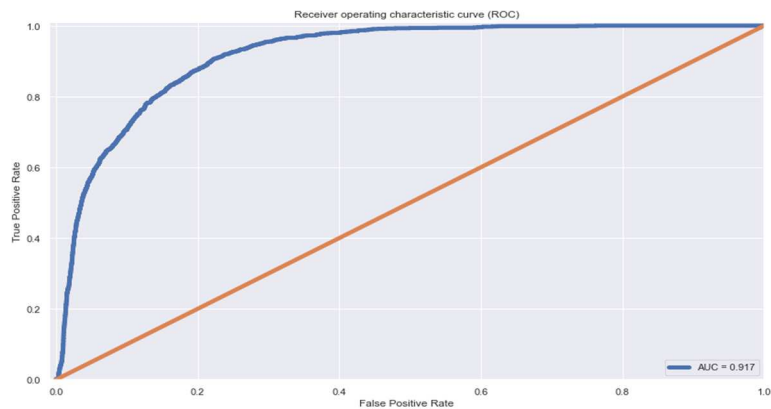


Figure 5-30. AUC of the AE model of threshold 1 built with dataset 2

The scores of the three-evaluation metrics of the AE model built with dataset 1 is presented in Table 5-34.

Threshold 1	Precision	Recall	F1-score
Dataset 2	0.78	0.09	0.16

Table 5-34. The results of evaluation metrics

All performances of the evaluation metrics increased and got better than the AE model built with dataset 2 using the threshold 4.

Lastly, Figure 5-31 illustrates the reconstruction error threshold of 1 in dataset 3. Narrowing down to only features that were likely to be effective in model discrimination made the thresholds of blue dots and orange dots clearer. A threshold value of 1 was more appropriate for the boundary between fraudulent data and non-fraudulent data, and the AE model discrimination accuracy was improved.

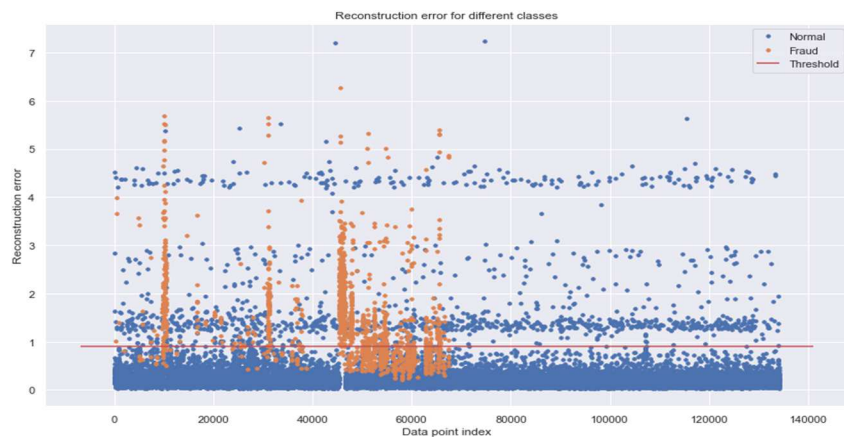


Figure 5-31. The reconstruction error threshold 1 in dataset 3

Figure 5-32 shows the confusion matrix of the AE model built with dataset 3 using the threshold of 1.

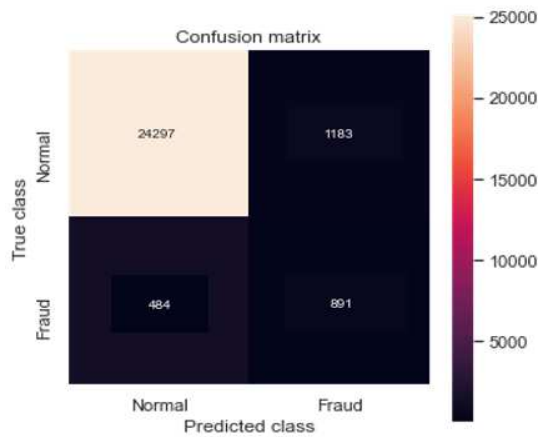


Figure 5-32. Confusion matrix of the AE model with dataset 3 using Threshold of 1

The AUC score became 0.96 as displayed in Figure 5-33 which is the best score amongst all AE models.

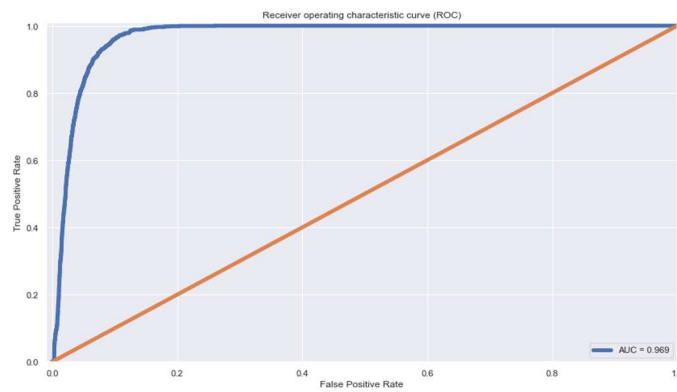


Figure 5-33. AUC of the AE model of threshold 1 built with dataset 3

The scores of the three-evaluation metrics of the AE model built with dataset 1 is shown in Table 5-35.

Threshold 1	Precision	Recall	F1-score
Dataset 3	0.43	0.65	0.52

Table 5-35. The results of evaluation metrics

## 5.6. Evaluation and Discussions

Through the whole experiment, performances of all models built with the dataset including the created features by using feature engineering methods in the framework was conspicuously improved than the models' performance when using only the original features. Here are individual assessments and a summary for each model's result. I summarised each score of four evaluation metrics: precision, recall, f1-score, and AUC, respectively for a better discussion. Recall measures the percentage of actual fraud transactions that were correctly classified. Precision measures the percentage of transactions flagged as fraud that were correctly classified. It all depends on what a user focus on. When maximising a precision score, the probability of judging the non-fraud transactions as fraudulent transaction can be lower. On the other hand, if a user wants to focus on detecting actual fraud transactions, then maximising a recall score is very important. F-1 score is a measure of combination of both precision and recall scores. F-1 score can help balance the metric across positive or negative samples. When evaluating the scores of each metrics, one needs to consider the balance of the two metrics of recall and precision.

### (A) Support Vector Machine models

Support vector machine was not good at handling a big data that is over 100,000 data. During the experiment, modelling and prediction took much time (i.e., over 6 hours) for obtaining only one model's result. Overall, it took over 18 hours for building three different SVM models and predicting test set.

Table 5-36 shows the summary of each score of the evaluation metrics in each model built with dataset 1, dataset 2, and dataset 3.

Dataset No.	Precision	Recall	F1-score	AUC
Dataset 1	0.00	0.03	0.00	0.09
Dataset 2	0.08	0.87	0.14	0.65
Dataset 3	0.19	0.28	0.23	0.61

*Table 5-36. The summary of all scores in the one-class SVM models*

Regarding the evaluation metrics' scores for the model trained on dataset 1, the precision score is 0.00, and recall score is 0.03. These low scores indicate that the model has not learned the different patterns between fraud and non-fraud transactions, which means that the model is unable to distinguish between fraudulent and non-fraudulent behaviour. Both scores of the models with dataset 2 and dataset 3 increased more than the scores of the model with dataset 1. It indicates that new features which were created by feature engineering methods could reveal the different pattern on transaction between a fraud and nonfraud. Comparing the model with dataset 2 and the model with dataset 3, specifically a recall of the model with dataset 2 is very higher than the model with dataset 3. On the other hand, the scores of a precision and F1-measure of the model with dataset 3 are higher than the model with dataset 2. A balance between precision and recall is a trade-off. If a bank places importance in the case of which customers are not classified as fraud, precision score should be considered preferentially. In this case, the dataset 3 will be the best features set for SVM. If a bank considers that detecting fraudulent cases are first priority, recall score will be significant and then, the dataset 2 will be the best features set for SVM. In any case, the total performance of the models built with dataset 2 and dataset 3 were significantly improved when comparing with the performance of the model built with dataset 1.

## (B) Random Forest models

Table 5-37 shows the summary of each score of the evaluation metrics in each model built with dataset 1, dataset 2, and dataset 3. As shown in Table 5-37, all model's performance is very high.

Dataset No.	Precision	Recall	F1-score	AUC
Dataset 1	1.0	0.48	0.64	0.74
Dataset 2	0.90	0.88	0.89	0.93
Dataset 3	0.99	0.88	0.93	0.94

*Table 5-37. The summary of all scores in the RF models*

Precision score in the model with dataset 1 became 1.0 which means that all customers could be classified as nonfraud. However, this model could not learn the patterns of fraudsters very well from the only raw features because the recall score became 0.48. On the other hand, both scores of precisions and recall in the models with dataset 2 and dataset 3 became much higher than the model with dataset 1. This result indicates that new features created by feature engineering methods could reveal the different patterns between customer and fraudsters. From all scores in the metrics, the models using dataset 2 and dataset 3 are greater than the model using dataset 1. This time, dataset 3 is suggested to use for the RF algorithm as the best combination of building a fraud detection model.

## (C) Isolation Forest models

Table 5-38 shows the summary of each score of the evaluation metrics in each model built with dataset 1, dataset 2, and dataset 3.

Dataset No.	Precision	Recall	F1-score	AUC
Dataset 1	0.19	0.16	0.17	0.55
Dataset 2	0.30	0.31	0.30	0.64
Dataset 3	0.34	0.36	0.35	0.66

Table 5-38. The summary of all scores in the IF models

Similar to previous observations, scores of precisions and recall in the models with dataset 2 and dataset 3 became almost doubled from the scores in the model with dataset 1. This indicates that more customers will be classified as nonfraud correctly and more fraudsters can be detected precisely by using the models built with the engineered features. Specifically, the model using dataset 3 surpassed the other IF models.

#### **(D) Isolation Outlier Factor models**

Table 5-39 shows the summary of each score of the evaluation metrics in each model built with dataset 1, dataset 2, and dataset 3.

Dataset No.	Precision	Recall	F1-score	AUC
Dataset 1	0.02	0.04	0.03	0.48
Dataset 2	0.05	0.05	0.05	0.50
Dataset 3	0.15	0.15	0.15	0.55

Table 5-39. The summary of all scores in the LOF models

Each score of performance measurements were very low in Table 5-39. This indicates that the LOF algorithm needs to be carefully set up with appropriate hyper parameters or trained in a different way while building the model. The important point of this research is to improve the baseline model's performance by using the new engineered features. Therefore, the model with dataset 1 was assumed as the baseline model and the performance of the models with dataset 2 and dataset 3 were more improved than the

model with dataset 1 even though the performance was not good with the LOF algorithm.

In this case, the best combination became the LOF algorithm and dataset 3.

### **(E) Autoencoder models**

The results in Table 5-40 show performance of the AE models using Threshold 4. Comparing the precision scores among all models, the models with dataset 2 and dataset 3 were improved drastically than the model with dataset 1. This indicates that almost all customers could be classified as nonfraud correctly. Focusing on the scores of recall, the AE model seems not to recognise the different pattern between customers and fraudsters.

Dataset No.	Precision	Recall	F1-score	AUC
Dataset 1	0.20	0.05	0.08	0.65
Dataset 2	0.93	0.07	0.13	0.83
Dataset 3	0.90	0.08	0.15	0.92

*Table 5-40. The summary of all scores in the AE models with Threshold 4*

In terms of the total results of the model's performance, both models built with dataset 2 and dataset 3 achieved higher scores than the performance of the model with dataset 1 in all respects. Regarding preventing money from being stolen by fraudsters, the best performance model is the AE model with dataset 3. However, the autoencoder does not only use input data to recognise the difference between fraud and nonfraud, but also it uses the threshold value for dividing by border between the fraud and nonfraud as shown in Table 5-41.

Next, Table 5-41 shows the results of the AE models using Threshold 1.



Dataset No.	Precision	Recall	F1-score	AUC
Dataset 1	0.19	0.06	0.09	0.77
Dataset 2	0.78	0.09	0.16	0.91
Dataset 3	0.43	0.65	0.52	0.96

*Table 5-41. The summary of all scores in the AE models with Threshold 1*

All performance of the AE models with Threshold 1 were improved more than the performance of the AE models with Threshold 4. This result indicates that using the effective dataset is to have an impact on the AE model's performance. And also, it is important to set an appropriate threshold value for improving the model's performance. Regarding the precision of the models with threshold 1, the scores were lower than the scores of all models with threshold 4 whereas the recall, AUC and F1-scores became higher. This result shows that the models have become more rigorous in detecting fraudsters and have determined that certain customers are fraudsters. Although the precision of the model with dataset 3 became lowest, the other metrics shows that the model could discriminate more clearly between customers and fraudsters.

Throughout the whole experiment, performances of all models built with dataset 2 and dataset 3 which include many new features created via the feature engineering steps in the framework were significantly improved than performances of the models built with dataset 1 which is an original dataset. Comparing the model with Dataset 2 and the model with dataset 3, there are a few different results between these models. First, the models built with dataset 3: Random Forest and Isolation Forest, were the best performance models among other same algorithm's models. These algorithms are based on the decision tree algorithm. So, this kind of algorithms based the decision tree algorithm will be improved the model's accuracy by using dataset 3 in which only effective candidate

features are selected. On the other hand, evaluation of the results of some algorithms such as one-class SVM, local outlier factor, and autoencoder depends on what impact a user cares about. Recall has an impact on huge money loss whereas precision influence on customer satisfaction and confidence. The score of precision shows the number of predictions as fraud where actual result is customer whereas the score of recall shows that the model predicts fraud transactions where actual result is fraud. A balance between precision and recall is a trade-off. Therefore, a user can select an appropriate dataset according to a purpose of building a model.

In conclusion, using the dataset containing only effective features will make machine learning and deep learning algorithms easier to identify fraudulent patterns than using the dataset containing both many effective and meaningless features. This indicates that irrelevant features will make it difficult to distinguish between fraudulent and non-fraudulent transactions while the algorithms are learning data patterns.

## **5.7. Conclusion**

In many financial fraud detection cases and studies, they used a variety of machine learning algorithms and deep learning. They suggested to use the different algorithms for financial fraud detections as shown in Chapter 2. Thus, I proposed a new feature engineering framework that can provide the most effective features set for any algorithms for financial fraud detection. Through the experiment, I proved the effectiveness of using the dataset provided by the framework. Furthermore, I also proved that there was compatibility between a given features set and a specific algorithm.

## **6. Conclusion and Future Work**

### **6.1. Introduction of the main achieved work**

This research seeks to advance the field of financial fraud detection by creating a new feature engineering framework consisting of two components: feature creation and selection. The approach adopted was to design and implement this framework by prototyping in order to provide evidence on the effectiveness and feasibility of this framework. Introduction of the fraud detection models built with the feature engineering dataset prepared by the framework demonstrated its advantage over several other baseline models built with the raw dataset. Introduction of multi-techniques of feature engineering that allows the creation of temporal action features and statistical and arithmetic features, which make machine learning and deep learning algorithms more easily to discover the different data patterns between normal and fraudulent transactions. In this chapter, this study will be concluded and will provide a closing remark on the problem statement as well as the effectiveness of the proposed solution. Furthermore, recommendations and future research areas that can improve the fraud detection model's performance are discussed as well.

As online payment system advances, fraud schemes have shifted from physical fraud actions by using the stolen credit card physically and directly at ATM or shops into online banking fraud actions by using advanced digital techniques in the internet websites. There are limitations of fraud detection system using only machine learning techniques for online banking transaction. Because of this there is an increase in fraudulent activities. Fraudsters no longer have to walk into a shop, ATM or bank's branch to carry out an attack, they can simply use someone's identity or steal the information through means of

hacking. The damage caused by these fraudulent activities goes beyond direct monetary loss for any financial institutions.

As discussed in Chapter 2 many of the existing studies of machine learning and deep learning algorithms are used for fraud detection cases, their limitations are compounded by fraudster's ability to continuously change their tactics in order to avoid detection. Only focusing on the improvement of machine learning techniques will not be able to catch up with these changing tactics and, moreover, as was explained in Chapter 2, there are not many existing studies which focus on feature engineering techniques for financial fraud detection. Research carried out in this study provide a proof of my hypothesis that the performance of fraud detection models can indeed be improved if machine learning or deep learning algorithms are trained by using the optimised feature set that contains new created features by various feature engineering techniques based on real life banking transactional data. After several theoretical analysis and analytical methods being described in Chapter 3, the output of the framework is summarised individually in different phases that eventually lead to producing the effective feature set for a fraud detection model. Figure 6-1 below shows the outputs in each different phase.

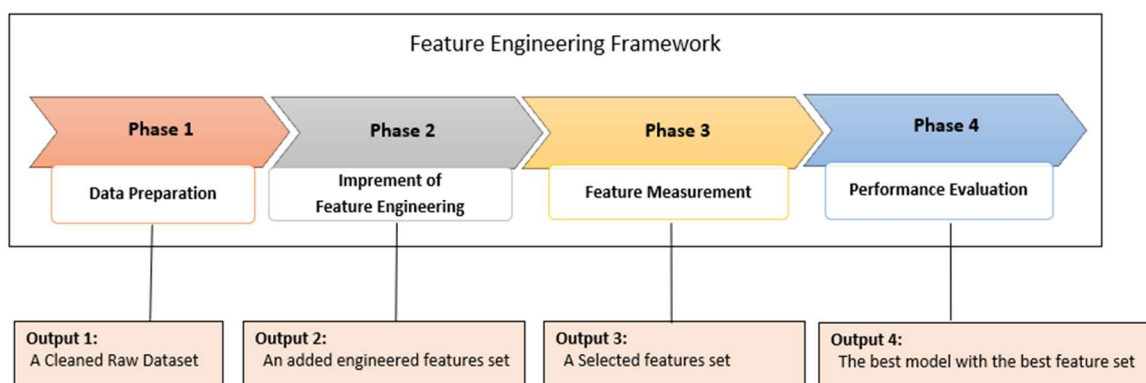


Figure 6-1. The output in each phase

**Phase 1 includes:**

- Data integration, modelling, and preparation processes. The quality of the whole dataset is determined based on the appropriate data extraction and processing methods in this phase.
- Extraction of required attributes from the integrated transaction data, which is used for feature engineering in Phase 2 to create new aggregated and transformed features. In general, the objects of attributes in banking transaction data are common, so I can define the feasible plan to extract the fixed features.
- Exploratory data analysis that can provide insight into hidden data patterns between normal and fraudulent transactions.
- Output that becomes the cleaned raw dataset, which can be used for a baseline model.

**Phase 2 includes:**

- Implementation of feature engineering methods: feature aggregation which creates new features based on customer's transaction behaviour by aggregating multiple attributes in the dataset prepared in Chapter 5, and feature transformation which creates new features based on mathematical and statistical functions.
- Feature aggregation methods: there are some common attributes which certainly exist in online banking data such as time, amount, balance, Internet information, customer's information, and event. In the framework, 35 aggregated features are

created based on customer's journeys on transaction at least if the expected attributes can be fully extracted.

- Feature transformation methods: there are some functions that are commonly utilised for feature transformations such as log transformation, counts, statistical functions, standardisation, PCA and so on. The purpose of feature transformation here is to create new features that can represent the latent data patterns which make a machine learning algorithm easily understand the difference between legitimate and fraud. In the framework, five mathematical functions which are popularly used in various research as feature transformation are adopted.

**Phase 3 includes:**

- Measurement for correlation coefficient values: The correlation coefficient is a statistical method to measure the degree of intensity of the relationship between two feature variables X and Y. This is used for avoiding a cause of overfitting.
- Feature importance measurement: The feature importance scores are calculated based on the information with regard to how many times each feature in training data contributes to model's discrimination.
- With regard to the results of correlation coefficient and feature importance, I selected the effective candidate features from all features in the dataset containing both the raw attributes and all created features in phase 2.

**Phase 4 includes:**

- Details on design and implementation of using various algorithms.

- Model evaluations by using predefined performance metrics such as precision, recall, AUC, and F1-score.
- The effective candidate features set that were created in phase 3 and were tested and evaluated based on appropriate performance metrics in the framework.
- The most effective features set can be provided for a specific model.

The design and implementation defined in the above phases was used to experiment the performance of three different types of machine learning namely SVM, isolation forests, local outlier factor, and an autoencoder as deep learning, using three different types of datasets explained in detail in Chapter 5. The best performance model was the RF model built with the selected features set in all respects when compared with performances of other models. Furthermore, performances of all algorithms using the dataset including new created features were dramatically improved. The results proved that use of the optimised features set provided by the feature engineering framework can improve ML/DL classifiers with better accuracy when compared to the use of the raw dataset for classifiers.

Overall, Table 6-1 below provides an overview on how the aim and objectives proposed in Chapter 1 are addressed.

Aim and Objectives	Status	Summary
To explore the current state of research in fraud detection and the cases specifically using feature engineering methods for classification models, and to identify the main issues, existing approaches, and available methods for improving performance of the fraud detection models.	Met	Details can be found in Chapter 2. Existing research covers across machine learning and deep learning models for fraud detection to understand current conditions, problems, and limitations. Regarding feature engineering, not a lot of research has been done on using feature engineering concepts to the dataset for classification in fraud detection. Moreover, very few papers exist on using both concepts of feature creation and selection simultaneously. Each method of feature engineering and selection for classifiers was investigated and summarised individually.
To investigate database structure tables of banking transaction and to consider which attributes in each table are constantly available to be extracted.	Met	Further details in Chapter 3. The main purpose of this aim and objective is to maintain a certain positive effect of using the framework. Therefore, mandatory attributes in any online banking transaction data are fixed. Even there is not full attributes in actual banking dataset, some key engineered features can be created with other mandatory attributes.
To investigate how to deal with character string datatype values and missing values in each attribute.	Partially Met	Further details can be found in Chapter 3. In the chapter, the general outline on how to deal with missing values and character string datatype is described, but part of the implementation is left for future work as it is beyond the scope of this research.
To research into both methods of feature engineering and selection for fraud detection and to consider how to create new features that express customer's behaviour during a transaction and reveal the different aspect of input values for making machine learning or deep learning models distinguish between normal and fraud easier. Also, to consider how to select the effective features from all attributes.	Partially Met	Further details can be found in Chapter 2 and Chapter 3. The thesis used several major techniques of feature engineering in various latest papers and studies for classification and applied them to financial fraud detection. Regarding feature selection, this time the effectiveness of using feature selection after feature creation is proved. However, in future work, there is still room for improvements on feature selection methods.
To analyse the multidimensional banking dataset which was provide by a private European in terms of both the exploratory data analysis with visualisation and the assessment of available attributes in the dataset.	Met	Further details in Chapter 4. Through the exploratory data analysis (EDA), trends and patterns in the dataset are investigated and provides insight of the data and the difference between normal and fraudulent transactions.

Table 6-1. An overview of the aim and objectives achieved



## 6.2. Research Contributions

This section gives a list of contributions as outcomes to this research. The core contribution of this research is to incorporate feature creation and feature selection into the framework and to provide the most effective features set for machine learning and deep learning algorithms. Specifically, in the feature creation process, both techniques of feature aggregation and feature transformation are included. Feature transformation methods are commonly used for deep learning in image recognition. In terms of creating new features with different aspects of the data, my research has combined the two concepts of feature aggregation and feature transformation and succeeded in producing brand new valuable features. Another novelty in the research is that many similar studies in financial sector have not done both feature engineering and feature selection consistently. Through the experiment, the effectiveness of using the features set generated by all processes in the framework was proven.

The heart of the concept consists of the following groups:

### (A) Novelty framework

- Advanced feature engineering technique that combines two different approaches: feature aggregation for expressing customer's behaviour during transaction, and feature transformation for mapping raw data into a different space that can reveal the latent pattern of the data into the framework.
- Combined both concepts of feature engineering and feature selection consistently.

As clearly shown the results in Chapter 5, the effectiveness of the integration of feature engineering and feature selection can be proved.

## (B) Improved methods for data preparation

- Data preparation that includes the significant ways how to deal with missing values and how to convert character string data into numerical data correctly. In the data preparation phase, the conceptual and actual methods of dealing with the missing or character string data are provided.

## 6.3. Recommendation for Future Research

The feature engineering framework that is proposed in this thesis can be used for machine learning and deep learning algorithms in financial fraud detection, specifically recommended for being used in online banking. However, collecting the overall network information and accessing device information into one place are challenging from a viewpoint of network security or private information.

Apart from the above consideration, the feature engineering framework can be further improved by:

**Obtaining a dataset having less loss values :** although the actual online banking transaction dataset was provided and had a large volume of records, many loss values in the dataset remained and included timestamp as well. If there were the full records of timestamp, other deep learning such as recurrent neural network (RNN) and convolutional neural network (CNN) could have been used and tested with the proposed features set.

**Implementing other techniques of feature selection:** In this thesis, fundamental concepts and techniques for measuring and selecting features were adopted. Through

using the more advanced methods of feature selection in the framework, the output features set can improve the performance of fraud detection models.

**Implementing other techniques of feature selection:** In this thesis, fundamental concepts and techniques for measuring and selecting features were adopted. Through using the more advanced methods of feature selection in the framework, the output features set can improve the performance of fraud detection models.

**Adding additional attributes related to customer's actions using credit card, online payment, and e-shops:** Feature engineering has still a high potential to create effective features for machine learning and deep learning algorithms. Adding features related to the customer's actions with use of credit card or shopping behaviours can create better engineered features in the framework and can make a better prediction model.

## References

1. **UK Finance Fraud Action** (2021) FRAUD – THE FACTS 2021: The definitive overview of payment industry fraud.
2. **Angela Makolo and Tayo Adeboye I.J.** (2021), ‘Credit Card Fraud Detection System Using Machine Learning’, *Information Technology and Computer Science*, Issue 4, P24-37, Published Online August 2021 in MECS, DOI: 10.5815/ijitcs.2021.04.03.
3. **Niccolo Mejia** (2020), AI-Based Fraud Detection in Banking – Current Applications and Trends [Online]. Available at: <https://emerj.com/ai-sector-overviews/artificial-intelligence-fraud-banking/> (Accessed: May 2021)
4. **Djeffal Abdelhamid, Soltani Khaoula, Ouassaf Atika** (2014), ‘Automatic Bank Fraud Detection Using Support Vector Machines’, *Proceedings of the International conference on Computing Technology and Information Management*, Dubai, UAE, 2014 ISBN: 978-0-9891305-5-4 ©2014 SDIWC
5. **Juergen Schmidhuber** (2015), ‘Deep Learning in Neural Networks: An Overview’, *Neural and Evolutionary Computing Journal*, vol. 61: P85–117, DOI: 10.1016/j.neunet.2014.09.003
6. **Emin Aleskerov, Bernd Freisleben, R. Rao.** (1997), ‘CARDWATCH: a neural network-based database mining system for credit card fraud detection’, *Proceedings of the IEEE/IAFE 1997 Computational Intelligence for Financial Engineering (CIFER)* (1997): P220-226.
7. **Marjan Abdeyazdan, Ali Rayat Pisheh** (2016), Discrimination Aware Decision Tree Learning [Online]. Available at: [http://iieng.org/images/proceedings\\_pdf/E0816003.pdf/](http://iieng.org/images/proceedings_pdf/E0816003.pdf/) (Accessed: Aug 2021)
8. **Chengwei Liu, Yixiang Chan, Syed Hasnain, Alam Kazmi, Hao Fu** (2017), Financial Fraud Detection Model: Based on Random Forest [Online]. Available at: <http://www.ccsenet.org/journal/index.php/ijef/article/viewFile/46957/27054> (Accessed: June 2021)
9. **Joshi, Shrijit, Phoha, Vir** (2005), ‘Investigating hidden Markov models capabilities in anomaly detection’, *Proceedings of the Annual Southeast Conference*, Vol 1. 98-103. 10.1145/1167350.1167387
10. **Foo Chi Hui, Venkaiah Chowdary Koneru, Norazman Mat Ali, Safurah Harun** (2014), ‘Implementing Peer Group Analysis within a Track and Trace System to Detect Potential Frauds’, *International Journal of Supply Chain Management*, Vol 3. No.1, P2051-30771
11. **Yoshihiro Ando, Hidehito Gomi, Hidehiko Tanaka** (2016), ‘Detecting Fraudulent Behaviour Using Recurrent Neural Networks’, *Computer Security Symposium*, Oct 2016
12. **Shuhao Wang, Cancheng Liu, Xiang Gao, Hongtao Qu, and Wei Xu** (2017), ‘Session-Based Fraud Detection in Online E-Commerce Transactions Using Recurrent Neural Networks’, *Machine Learning and Knowledge Discovery in Databases*, Springer International Publishing 2017

13. **Apapan Pumsirirat and Liu Yan** (2018), 'Credit Card Fraud Detection using Deep Learning based on Auto-Encoder and Restricted Boltzmann', *International Journal of Advanced Computer Science and Applications (ijacsa)*, Vol 9, Issue 1, DOI:/10.14569/IJACSA.2018.090103
14. **Isa Modibbo Ismail and Ekpe Okorafor** (2021), An adaptive predictive financial fraud detection approach using deep learning methods on a big data platform [Online]. Available at: <https://afribary.com/works/an-adaptive-predictive-financial-fraud-detection-approach-using-deep-learning-methods-on-a-big-data-platform> (Accessed: June 2021)
15. **Yibo Wang, Wei Xu** (2018), 'Leveraging deep learning with LDA-based text analytics to detect automobile insurance fraud', *Decision Support Systems*, Volume 105, P 87-95
16. **Sam Scott, Stan Matwin** (1999), 'Feature Engineering for Text Classification', *Proceedings of ICML-99, 16th International Conference on Machine Learning*, P 379-388
17. **Alice X. Zheng** (2017), *Mastering Feature Engineering: Principles and Techniques for Data Scientists* [online]. Available at: [https://www.repath.in/gallery/feature\\_engineering\\_for\\_machine\\_learning.pdf](https://www.repath.in/gallery/feature_engineering_for_machine_learning.pdf) (Accessed: January 2022)
18. **Rakhi Chakraborty** (2013), *Domain Keyword Extraction Technique : A New Weighting Method based on Frequency Analysis* [online]. Available at: DOI:10.5121/CSIT.2013.3211 (Accessed: Sep 2020)
19. **Gilad Katz, E.C.Richard Shin, Dawn Song** (2016), 'ExploreKit: Automatic Feature Generation and Selection', *16th IEEE International Conference on Data Mining, ICDM 2016*, P979-984. 10.1109/ICDM.2016.0123
20. **James Max Kanter, Kalyan Veeramachaneni** (2015), *Deep Feature Synthesis: Towards Automating Data Science Endeavors*, 2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA), pp. 1-10, DOI: 10.1109/DSAA.2015.7344858
21. **James Max Kanter, Kalyan Veeramachaneni** (2016), *Label, Segment, Featurise: a cross domain framework for prediction engineering*, In 2016 IEEE International Conference on Data Science and Advanced Analytics, DSAA 2016, P 430-439
22. **Pawel Grabinski** (2018), *Feature Engineering for Machine Learning: 10 Examples, KDnuggets* [online]. Available at: <https://www.kdnuggets.com/2018/12/feature-engineering-explained.html> (Accessed: June 2021)
23. **Daniel Massa, Raul Valverde** (2014), 'A Fraud Detection System Based on Anomaly Intrusion Detection Systems for E-Commerce Applications', *Journal Computer Information Science* Vol.7, No.2, DOI:10.5539/cis.v7n2p117
24. **Ogwueleka, Francisca** (2011), 'Data mining application in credit card fraud detection system', *Journal of Engineering Science and Technology*. Vol.6. P311-322, 2011
25. **Rakhi Chakraborty** (2013), 'Domain Keyword Extraction Technique: A New Weighting Method Based on Frequency Analysis', *ACER 2013*, pp. 109118, DOI :

26. **Fabrizio Carcillo, Andrea Dal Pozzolo, Yann-Aël Le Borgne, Olivier Caelen, Yannis Mazzer, Gianluca Bontempi** (2018), 'SCARFF: a Scalable Framework for Streaming Credit Card Fraud Detection with Spark', *Journal Information Fusion* 41C, p182-194, Available at: <https://doi.org/10.1016/j.inffus.2017.09.005>
27. **Dongfang Zhang, Basu Bhandari, Dennis Black** (2020), 'Credit Card Fraud Detection Using Weighted Support Vector Machine', *Applied Mathematics in Model Development Department, Comerica Bank* [online]. Available at: DOI: 10.4236/am.2020.1112087, (Accessed: December 2020)
28. **V. Dheepa, R. Dhanapal** (2012), 'Behaviour based credit card fraud detection using support vector machines', *Journal SOCO 2012*, DOI:10.21917/IJSC.2012.0061
29. **Ifedayo Oladeji, Peter Makolo, Ramon Zamora, Tek Tjing Lie** (2021), 'Density-based clustering and probabilistic classification for integrated transmission-distribution network security state prediction', *Journal of WILEY*, Volume 211, DOI:10.1002/widm.1342
30. **E.A. Amusan O.M. Alade O.D. Fenwa J.O. Emuoyibofarhe** (2021), 'Credit Card Fraud Detection on Skewed Data using Machine Learning Techniques', *Journal of Computing and Informatics (LAUJCI) – ISSN: 2714-4194 Volume 2 Issue 1*
31. **Pooja Tiwari, Simran Mehta, Nishtha Sakhuja, Jitendra Kumar, Ashutosh Kumar Singh** (2021), *Credit Card Fraud Detection using Machine Learning, Artificial Intelligence* [online]. Available at: arXiv:2108.10005 (Accessed: Aug 2021)
32. **M. S. Kumar, V. Soundarya, S. Kavitha, E. S. Keerthika and E. Aswini** (2019), 'Credit Card Fraud Detection Using Random Forest Algorithm', *IEEE Conference Publication, 2019 3rd International Conference on Computing and Communications Technologies (ICCCT)*, pp. 149-153, DOI: 10.1109/ICCCT2.2019.8824930
33. **R. Sailusha, V. Gnaneswar, R. Ramesh and G. R. Rao** (2020), 'Credit Card Fraud Detection Using Machine Learning', *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)*, P1264-1270, DOI: 10.1109/ICICCS48265.2020.9121114
34. **Casilda Aresti** (2018), *Technology and operations management: PayPal's Use of Machine Learning to Enhance Fraud Detection* [online]. Available at: <https://digital.hbs.edu/platform-rctom/submission/paypals-use-of-machine-learning-to-enhance-fraud-detection-and-more/> (Accessed: Nov 2019)
35. **Raghavendra Patidar, Lokesh Sharma** (2011), 'Credit Card Fraud Detection Using Neural Network', *India International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Vol. 1, Issue-NCAI2011*
36. **Morteza Kolali Khormuji, Mehrnoosh Bazrafkan, Maryam Sharifian, Seyed Javad Mirabedini, Ali Harounabadi** (2014), 'Credit Card Fraud Detection with a Cascade Artificial Neural Network and Imperialist Competitive Algorithm', *International Journal of Computer Applications (0975 8887) Vol. 96 - No. 25*
37. **KolaliKhormuji, Morteza, Bazrafkan, Mehrnoosh, Sharifian, Maryam, Mirabedini, Seyed, Harounabadi Ali** (2014), 'Credit Card Fraud Detection with a

- Cascade Artificial Neural Network and Imperialist Competitive Algorithm’, *International Journal of Computer Applications*, Vol. 96 P 1-9, DOI: 10.5120/16947-6736
38. **Parvinder Singh, Mandeep Singh** (2015), ‘Froud Detection by Monitoring Customer Behaviour and Activities’, *International Journal of Computer Applications*, Vol. 111, No 11
  39. **Mohamed Hegazy, Ahmed Madian, Mohamed Ragaie** (2016), ‘Enhanced Fraud Miner: Credit Card Fraud Detection using Clustering Data Mining Techniques’, *Egyptian Computer Science Journal* (ISSN: 1110 – 2586) Volume 40 – Issue 03
  40. **B. Angelin and A. Geetha** (2020), "Outlier Detection using Clustering Techniques – K-means and K-median," 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS), P373-378, DOI: 10.1109/ICICCS48265.2020.9120990
  41. **B.A. Abdulsalami, A. A. Kolawole, M.A. Ogunrinde, M. Lawal, R.A. Azeez, A.Z. Afolabi** (2019), ‘Comparative Analysis of Back-propagation Neural Network and K-Means Clustering Algorithm in Fraud Detection in Online Credit Card Transactions’, *Fountain Journal of National and Applied Science*, Available at: <https://doi.org/10.53704/fujnas.v8i1.315> (Accessed: Jun 29, 2019)
  42. Isolation Forest and Local Outlier Factor for Credit Card Fraud Detection System, **V. Vijayakumar, Nallam Sri Divya, P. Sarojini, K. Sonika**, *International Journal of Engineering and Advanced Technology (IJEAT)*, ISSN: 2249 – 8958, Volume-9 Issue-4, April 2020
  43. Using isolation forest in anomaly detection: The case of credit card transactions, **Soumaya Ounacer, Hicham Ait El Bour, Younes Oubrahim, Mohamed Yassine Ghomari, Mohamed Azzouazi**, Vol 6, No 2, 2018
  44. A Hybrid and Improved Isolation Forest Algorithm for Anomaly Detection, **G. Madhukar RaoDharavath Ramesh**, *International Conference on Recent Trends in Machine Learning, IoT, Smart Cities and Applications* pp 589-598, October 2020
  45. **Topics, G. Kumar Singh, A. Bhayye, S. Dhamnaskar, S. Patil, and S. V. Phulari** (2021), ‘Credit Card Fraud Detection Using Isolation Forest’, *International Journal of Recent Advances in Multidisciplinary, IJRAMT*, vol. 2, no. 6, P118–119
  46. **Hyder John, Sameena Naaz** (2019), ‘Credit Card Fraud Detection using Local Outlier Factor and Isolation Forest’, *International Journal of Computer Science and Engineering* vol.7, Issure4, P1060-1064, DOI: 10.26438/ijcse/v7i4.10601064
  47. **Shubham Jaiswal, R. Brindha, Shubham Lakhotia** (2021), ‘Credit Card Fraud Detection Using Isolation Forest and Local Outlier Factor’, *Annals of R.S.C.B.*, ISSN: 1583-6258, Vol. 25, Issue 5, 2021, Pages. 4391 – 4396
  48. **Diwakar Tripathi, Tushar Lone, Yograj Sharma** (2018), ‘Credit Card Fraud Detection using Local Outlier Factor’ , *International Journal of Pure and Applied Mathematics*, Vol. 118, No.7, P229-234
  49. **Jeremy Jordan** (2018), Introduction to autoencoders [Online]. Available at: <https://www.jeremyjordan.me/autoencoders> (Accessed at: March 2019)

50. **Zou Junyi, Jinliang Zhang, Pin Jiang** (2019), 'Credit Card Fraud Detection Using Autoencoder Neural Network', Journal of ArXiv, abs/1908.11553
51. **Misra, Sumit, Soumyadeep Thakur, Manosij Ghosh, Sanjoy Kumar Saha** (2020), 'An Autoencoder Based Model for Detecting Fraudulent Credit Card Transaction', Procedia Computer Science Vol. 167, P 254-262, 2020
52. **Apapan Pumsirirat, Liu Yan** (2018), 'Credit Card Fraud Detection using Deep Learning based on Auto-Encoder and Restricted Boltzmann Machine', International Journal of Advanced Computer Science and Applications(IJACSA), Volume 9 Issue 1
53. Data Camp: Understanding Random Forests Classifiers in Python Tutorial [online] <https://www.datacamp.com/tutorial/random-forests-classifier-python> (Accessed: Dec 2021)
54. **Alom Md. Zahangir, Taha Tarek, Yakopcic Chris, Westberg Stefan, Sidike Paheding, Nasrin Mst, Hasan Mahmudul, Essen Brian, Awwal Abdul** (2019), 'A State-of-the-Art Survey on Deep Learning Theory and Architectures', Asari Vijayan Electronics. Vol 8. DOI:292. 10.3390/electronics8030292
55. **F. Milletari, N. Navab, S. Ahmadi** (2016), 'V-Net: Fully Convolutional Neural Networks for Volumetric Medical Image Segmentation', Fourth International Conference on 3D Vision (3DV), Stanford, CA, USA, P565-571
56. **Maturana and Scherer** (2015), 'VoxNet: A 3D Convolutional Neural Networks for Real-Time Object Recognition', 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), P922-928, DOI: 10.1109/IROS.2015.7353481
57. **Zhaohui Zhang, Xinxin Zhou, Xiaobo Zhang, Lizhi Wang, Pengwei Wang** (2018), 'A Model Based on Convolutional Neural Network for Online Transaction Fraud Detection', Security and Communication Networks, vol. 2018, Article ID 5680264, P 9
58. **Mohsen Hadian, Seyed Mohammad Ebrahimi Saryazdi, Ardashir Mohammadzadeh, Masoud Babaei**, (2021), 'Chapter 11 - Application of artificial intelligence in modelling, control, and fault diagnosis', Applications of Artificial Intelligence in Process Systems Engineering', Application of Artificial Intelligence in Process Systems Engineering, P255-323, DOI: 10.1016/B978-0-12-821092-5.00006-1
59. **Hochreiter and Schmidhuber** (1997), 'Long short-term memory', Neural Computation, Vol 9, Issue 8, pp. 1735-1780, DOI: 10.1162/neco.1997.9.8.1735
60. **Ihianle Isibor, Nwajana Augustine, Eбенуwa Solomon, Otuka Richard , Owa Kayode, Orisatoki Mobolaji** (2020), 'A Deep Learning Approach for Human Activities Recognition from Multimodal Sensing Devices', IEEE Access. Vol 8. 179028-179038. 10.1109/ACCESS.2020.3027979
61. **Antonio Martini Barclays** (2022), Deep Recurrent Neural Networks for Fraud Detection on Debit Card Transactions: Quantitative Analytics, Fraud Detection 2022 [online]. Available at: <https://www.crc.business-school.ed.ac.uk/sites/crc/files/2020->



10/E29-Deep-Recurrent-Neural-Networks-Martini.pdf (Accessed: June 2021)

62. **Ibtissam Benchaji, Samira Douzi, and Bouabid El Ouahidi, Mohammed V, Rabat** (2021), 'Credit Card Fraud Detection Model Based on LSTM Recurrent Neural Networks', *Journal of Advances in Information Technology* Vol. 12, No. 2
63. **K. Fu, D. Cheng, Y. Tu, and L. Zhang** (2017), 'Credit Card Fraud Detection Using Convolutional Neural Networks', *ICONIP 2016*, DOI:10.1007/978-3-319-46675-0\_53
- 64., **S. Y. Huang, R. H. Tsaih, and W. Y Lin** (2014), Feature Extraction of Fraudulent Financial Reporting Through Unsupervised Neural Networks [Online]. Available at: <http://www.nnw.cz/doi/2014/NNW.2014.24.031.pdf> (Accessed: Jun 2020)
65. **Yvan Lucas, Pierre-Edouard Portier, Léa Laporte, Liyun He-Guelton, Olivier Caelen** (2019), 'Towards automated feature engineering for credit card fraud detection using multi-perspective HMMs', *Future Generation Computer Systems*, Vol.102, P 393-402, ISSN 0167-739X, DOI:10.1016/j.future.2019.08.029
66. **Xinwei Zhang, Yaoci Han, Wei Xu, Qili Wang** (2021), 'HOBA: A novel feature engineering methodology for credit card fraud detection with a deep learning architecture', *Information Sciences*, Vol.557, P302-P316, ISSN 0020-0255
67. **Alejandro Correa Bahnsen, Djamila Aouada, Aleksandar Stojanovic, Björn Ottersten** (2018), 'Feature engineering for credit card fraud detection', *Expert Systems with Applications*, Vol.51, P134-142, ISSN 0957-4174, DOI:10.1016/j.eswa.2015.12.030
68. **Nagaraja Arun, B.Uma, Khalaf Khatatneh, Radhakrishna Vangipuram, N.Rajasekhar, Kiran V.Sravan** (2020), 'Similarity based feature transformation for network anomaly detection', *IEEE Access*. P11. DOI:10.1109/ACCESS.2020.2975716
69. **Jeff Heaton** (2017), Nova South-eastern University :Automated Feature Engineering for Deep Neural Networks with Genetic Programming at College of Computing and Engineering [online]. Available at: [https://www.heatonresearch.com/dload/phd/jheaton\\_dissertation\\_10259604.pdf](https://www.heatonresearch.com/dload/phd/jheaton_dissertation_10259604.pdf) (Accessed: Sep 2020)
70. **F. Nargesian, H. Samulowitz, U. Khurana, E. B. Khalil, Deepak Turaga** (2016), 'Learning Feature Engineering for Classification', *The 26<sup>th</sup> International Joint Conference on Artificial Intelligence*, P2529-2535, DOI:10.24963/ijcai.2017/352
71. **James Max Kanter, Kalyan Veeramachaneni** (2016), 'Label, Segment, Features: a cross domain framework for prediction engineering', *2016 IEEE International Conference on Data Science and Advanced Analytics*, P430-439, DOI: 10.1109/DSAA.2016.54
72. **Alina Raphael, Zvy Dubinsky, David Iluz, Nathan Netanyahu** (2020), 'Neural Network Recognition of Marine Benthos and Corals', *Conference: 1st International Electronic Conference on Biological Diversity, Ecology and Evolution*, DOI:10.3390/BDEE2021-09415
73. **Sebastian Raschka** (2014), About Feature Scaling and Normalization and the effect

of standardization for machine learning algorithms [Online]. Available at: [https://sebastianraschka.com/Articles/2014\\_about\\_feature\\_scaling.html](https://sebastianraschka.com/Articles/2014_about_feature_scaling.html) (Accessed: Jul 2020)

74. **Leo Breiman, Jerome H. Friedman** (1985), 'Estimating optimal transformations for multiple regression and correlation', *Journal of the American Statistical Association*, Vol. 80, P580-598 DOI:10.1080/01621459.1985.104781572017
75. **DeZyre Tutorials** (2018), Principal Component Analysis Tutorial [Online]. Available at: <https://www.dezyre.com/data-science-in-python-tutorial/principal-component-analysis-tutorial> (Accessed: May 2022)
76. **Analytics Vidhya Content Team** (2018), Analytics Vidhya :Practical Guide to Principal Component Analysis (PCA) in R and Python [Online]. Available at: <https://www.analyticsvidhya.com/blog/2016/03/practical-guide-principal-component-analysis-python> (Accessed: April 2020)
77. **Dheepa and R. Dhanapal** (2012), 'Behaviour based credit card fraud detection using support vector machines', *ICTACT Journal on Soft Computing*, Vol. 02, Issue 04, DOI: DOI: 10.21917/ijsc.2012.0061
78. **M. R. Lepoivre, C. Avanzini, G. Bignon, L. Legender, A. K. Piwele** (2016), 'Credit Card Fraud Detection with Unsupervised Algorithms' , *Journal of Advances in Information Technology* Vol. 7, No. 1, P34-38
79. **Roy Wedge, James M. Kanter, Kalyan V** (2017), Solving the "false positive" problem in fraud prediction, In book: *Machine Learning and Knowledge Discovery in Databases*, P372-388, DOI: 10.1007/978-3-030-10997-4\_23
80. **Dewang Nautiyal** (2022), ML|Underfitting and Overfitting, *Advanced Computer Subject* [online]. Available at: <https://en.matasaroja.com/notipuhu/underfitting-and-overfitting-in-machine-learning/?ref=leftbar-rightbar> (Accessed: Jun 2022)
81. Scikit-learn: Underfitting VS Overfitting, Scikit Learn Official site [online]. Available at: [https://scikit-learn.org/0.15/auto\\_examples/plot\\_underfitting\\_overfitting.html](https://scikit-learn.org/0.15/auto_examples/plot_underfitting_overfitting.html) (Accessed May 2022)
82. **Andrei Dmitri Gavrilov, Alex Jordache, Maya Vasdani, Jack Deng** (2018), 'Preventing Model Overfitting and Underfitting in Convolutional Neural Networks', *International Journal of Software Science and Computational Intelligence (IJSSCI)*, Vol 10. Issue 4, P18-28, DOI: 10.4018/IJSSCI.2018100102
83. **H. Zhang, L. Zhang and Y. Jiang** (2019), 'Overfitting and Underfitting Analysis for Deep Learning Based End-to-end Communication Systems', *11th International Conference on Wireless Communications and Signal Processing (WCSP)*, P1-6, DOI:10.1109/WCSP.2019.8927876
84. **R.C. Chen, S.T. Luo, X. Liang, V.C.S. Lee** (2021), Personalized approach based on SVM and ANN for detecting credit card fraud, *Proceedings of the IEEE International Conference on Neural Networks and Brain*. pp. 810-815
85. **Jennifer G. Dy, Carla E. Brodley** (2004), 'Feature selection for unsupervised learning', *Journal of Machine Learning Research*, Vol. 5, P144, DOI:10.5555/1005332.1016787

86. **Kajal Kamaljit Kaur** (2021), 'Credit Card Fraud Detection using Imbalance Resampling Method with Feature Selection', *International Journal of Advanced Trends in Computer Science and Engineering*, Vol. 10 No. 3, P2016-2071
87. **Zhaohui Zhang, Xinxin Zhou, Xiaobo Zhang, Lizhi Wang, Pengwei Wang** (2018), 'A Model Based on Convolutional Neural Network for Online Transaction Fraud Detection', *Security and Communication Networks*, vol. 2018, Article ID 5680264, P9
88. **Aurelien Geron** (2019), Chapter 1: Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow. [Textbook] O'RELLY, P16-17, 1<sup>st</sup> ED.
89. **Pier Paolo Ippolito** (2019), SVM: Feature Selection and Kernels, *Towards Data Science* [online]. Available at: <https://towardsdatascience.com/svm-feature-selection-and-kernels-840781cc1a6c> (Accessed: Nov 2021)
90. **Cornell University Computer Science** (2018) : Lecture 9: Support Vector Machine [online]. Available at: <https://www.cs.cornell.edu/courses/cs4780/2018fa/lectures/lecturenote09.html> (Accessed: Aug 2018)
91. **Diana Ramos, Smartsheet** (2018), Real-Life and Business Applications of Neural Networks [online]. Available at: <https://www.smartsheet.com/neural-network-applications> (Accessed: Aug 2022)
92. **NVIDIA** (2020), K-Means Clustering Algorithm: Glossary [online]. Available at: <https://www.nvidia.com/en-us/glossary/data-science/k-means/> (Accessed: Nov 2021)
93. **Wo-Ruo Chen, Yong-Huan Yun, Ming Wen, Hong-Mei Lu, Zhi-Min Zhang** (2016), 'Representative subset selection and outlier detection via isolation forest', *Analytical Methods*, Issue 39, 8. 10.1039/C6AY01574C. DOI:10.1039/C6AY01574C
94. **Recurrent Neural Networks-Remembering what's important** (2019), gotensor Recurrent Neural Network Article [online] <https://gotensor.com/2019/02/28/recurrent-neural-networks-remembering-whats-important/> (Accessed: Dec 2021)
95. **Robert Keim** (2019), How to Use a Simple Perceptron Neural Network Example to Classify Data Technical Article [online] <https://www.allaboutcircuits.com/technical-articles/how-to-perform-classification-using-a-neural-network-a-simple-perceptron-example/> (Accessed: Aug 2022)
96. **Jian Yang , Zixin Tang, Zhenkai Guan, Wenjia Hua, Mingyu Wei, Chunjie Wang, and Chenglong Gu**, 'Automatic Feature Engineering-Based Optimization Method for Car Loan Fraud Detection', *Discrete Dynamics in Nature and Society*, vol. 2021, Article ID 6077540, 10 pages, 2021. <https://doi.org/10.1155/2021/6077540>
97. **UK Area Codes**, Official website [Online]. Available at: <https://www.visitnorthwest.com/uk-area-codes/> (Accessed: July 2022)
98. How and where to apply Feature Scaling, Shaurya Uppal, Medium.com [online]. Available at: <https://shauryauppal.medium.com/how-and-where-to-apply-feature-scaling-machine-learning-93316663cd63> (Accessed: July 2022)
99. **Premanand S** (2021), The A-Z guide to Support Vector Machine, *Analytics Vidhya*

- [online] <https://www.analyticsvidhya.com/blog/2021/06/support-vector-machine-better-understanding/> (Accessed: June 2022)
100. **Siddharth Misra, Hao Li** (2020), Non-invasive fracture characterization based on the classification of sonic wave travel times, in Machine Learning for Subsurface Characterization, DOI:10.1016/b978-0-12-817736-5.00009-0
  - 101.1.4. Support Vector Machines, Scikit Learn Official [Online]. Available at: <https://scikit-learn.org/stable/modules/svm.html> (Accessed: Aug 2022)
  102. **Daniel Chepenko** (2018), A Density-based algorithm for outlier detection, Towards Data Science [online] <https://towardsdatascience.com/density-based-algorithm-for-outlier-detection-8f278d2f7983> (Accessed: Dec 2021)
  103. **Udayan Khurana, Horst Samulowitz, Deepak Turaga** (2018), 'Feature Engineering for Predictive Modelling Using Reinforcement Learning', The Thirty-Second AAAI Conference on Artificial Intelligence (AAAI-18), Vol. 32, No.1, DOI: 10.1609/aaai.v32i1.11678
  104. **Inna Logunova** (2022), Feature Engineering for Machine Learning, Serokell Labs [online] <https://serokell.io/blog/feature-engineering-for-machine-learning> (Accessed: Dec 2022)
  105. Data Flow Diagram for Online Banking System [Online]. Available at: Geeks for Geeks data <https://www.geeksforgeeks.org/data-flow-diagram-for-online-banking-system/> (Accessed: Dec 2021)
  106. **Andrea Trevino** (2016), Introduction to K-means Clustering, Oracle AI & Data Science [online] <https://blogs.oracle.com/ai-and-datascience/post/introduction-to-k-means-clustering> (Accessed: Dec 2022)
  107. **Zou, Kelly & Sidharthan, Shawn & DeTora, Lisa & Chen, Yunmei & Ragin, Ann & Edelman, Robert & Wu, Ying** (2010), Statistical Evaluations of the Reproducibility and Reliability of 3-Tesla High Resolution Magnetization Transfer Brain Images: A Pilot Study on Healthy Subjects. International journal of biomedical imaging. 2010. 618747. 10.1155/2010/618747.
  108. Splitting a Dataset into Train and Test Sets [Online] Available at: Baeldung <https://www.baeldung.com/cs/train-test-datasets-ratio> (Accessed: Dec 2022)
  109. Precision versus recall: Differences, Use Cases and Evaluation [Online] Available at: V7Labs <https://www.v7labs.com/blog/precision-vs-recall-guide>
  110. Precision-Recall curve and AUC-RP [Online] Available at: <https://hasty.ai/docs/mp-wiki/metrics/precision-recall-curve-and-auc-pr> (Dec 2022)

# Appendix

## Experimental Setup

### (a) Workspace

In the experiment, I installed Anaconda Navigator on my computer, set up a Python environment that has various common machine learning libraries, and configured Jupyter Notebook to implement the framework.

Python is a very productive programming language, and it provides extensive libraries which can be popularly and easily used for data analysis and machine learning and deep learning model development. Anaconda is an open-source Python distribution with many software tools such as Jupyter Notebook. Anaconda for Windows was installed from <https://www.anaconda.com/products/distribution> including Python 3.7 or higher and used its packaging system.

### (b) Python Libraries

There are several useful Python libraries: NumPy, pandas, Matplotlib, Scikit-Learn, TensorFlow. for analysing data and building machine learning models. Table 1 describes the summary of the libraries used in the experiment:

Library Names	Description
Numpy	Numpy stands for numerical python, and it is the commonly used for assisting large matrices and multi-dimensional data. It includes mathematical functions for easy computations. TensorFlow also uses Numpy inside to carry out multiple operations on tensors.
Pandas	Pandas is a significant library that provides various analysis tools for visualising, manipulating, and cleaning data. It supports operations such as aggregating, transforming, indexing, sorting, and converting data.
Scikit-learn	Scikit-learn is a valuable library to handle complex data and works in conjunction with Numpy. It supports machine learning algorithms which include a variety of supervised and unsupervised learning, e.g., classification, clustering, regression.
Matplotlib	Matplotlib is a useful library for plotting numerical data with graphs, pie charts, histograms, scatterplots, and so on.

TensorFlow	TensorFlow is an open-source library used for high-level computations. This library is also utilized in machine learning and deep learning algorithms with many tensor operations. It provides a solution for solving complex computations in Mathematics.
Keras	It makes possible to deal with deep learning engines easily like TensorFlow with Python.

Table 1. Python Common Libraries

### (c) Machine learning and Deep Learning algorithms

The above libraries provide several modules for creating machine learning and deep learning models. The five different modules were used for building the models. Before building each model, the dataset was split into training set and test set by using one of the python library in Table 2.

Methods	Hyperparameter	Modules
Train Test Split	test_size, random_state	from sklearn.model_selection import train_test_split Train_x, Text_x, Train_y, Test_y = train_test_split(X,Y)
One-Class SVM	kernel, degree, gamma, max_iter, random_state	from sklearn.svm import OneClassSVM clf = OneClassSVM ( )
Random Forest	max_depth, random_state	import sklearn.ensemble import RandomForestClassifier clf = RandomForestClassifier ( )
Isolation Forest	n_estimators max_samples contamination random_state	import sklearn.ensemble import IsolationForest clf = IsolationForest ( )
Local Outlier Factor	n_neighbors contamination	from sklearn.neighbors import LocalOutlierFactor clf = LocalOutlierFactor ( )
Autoencoder	optimizer metrics loss nb_epochs batch_size	from keras.models import Model, load_model from keras.layers import Input, Dense autoencoder.compile( )

Table 2. Python Modules of machine learning and deep learning

#### (d) Performance Evaluation Methods

The four metrics of model validation in this experiment were AUC, precision, recall, and F1-measure. These methods were provided in Scikit-Learn library as described in Table 3.

Methods	Modules
AUC	<pre>from sklearn.metrics import roc_auc_score roc_auc_score ( Train_y, y_scores)</pre>
Precision	<pre>from sklearn.metrics import precision_score precision_score (Train_y, Test_y)</pre>
Recall	<pre>from sklearn.metrics import recall_score recall_score (Train_y, Test_y)</pre>
F1-measure	<pre>from sklearn.metrics import f1_score f1_score (Train_y, Test_y)</pre>

*Table 3. Python Modules of Performance Evaluation Metrics*