Strathmore UNIVERSITY

Fraud Detection using Machine Leaning:
A Comparative Analysis of Neural Networks & Support Vector Machines

Gitonga Joseph Theuri

083568

Submitted in partial fulfilment of the requirements for the Degree of Bachelor of Business Science in Finance at Strathmore University

Strathmore Institute for Mathematical Sciences

Strathmore University

Nairobi, Kenya

November 17th 2017

DECLARATION

I declare that this work has not been previously submitted and approved for the award of a degree by this or any other University. To the best of my knowledge and belief, the Research Proposal contains no material previously published or written by another person except where due reference is made in the Research Proposal itself.

JOSEPH THEURI GITONGA. [*Name of Candidate*]

..... [*Signature*]

7TH DEC 2017. [*Date*]

This Research Proposal has been submitted for examination with my approval as the Supervisor.

Dr Okuku [*Name of Supervisor*]

..... [*Signature*]

7/12/2017 [*Date*]

..... [*Name of Supervisor*]

..... [*Signature*]

..... [*Date*]

Strathmore Institute of Mathematical Sciences

Strathmore University

Abstract

Fraud detection and prevention tools have been evolving over the past decade with the ever growing combination of resources, tools, and applications in big data analytics. The rapid adoption of a new breed of models is offering much deeper insights into data. There are numerous machine learning techniques in use today but irrespective of the method employed the objective remains to demonstrate comparable or better recognition performance in terms of the precision and recall metrics. This study evaluates two advanced Machine Learning approaches: Support Vector Machines and Neural Networks while taking a look at Deep Learning. The aim is to identify the approach that best identifies fraud cases and discuss challenges in their implementation. The approaches were evaluated on real-life credit card transaction data. Support Vector Machines demonstrated overall better performance across the various evaluation measures although Deep Neural Networks showed impressive results with better computational efficiency.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

ANN	-	Artificial Neural Network
CVM	-	Core Vector Machine
DNN	-	Deep Neural Networks
MLP	-	Multilayer Perceptron
MMSP	-	Mobile Money Service Providers
NN	-	Neural Networks
TCP	-	Protocol
IP	-	Internet Protocol
SOFM	-	Self-Organizing Feature Map
FFNN	-	Feed Forward Neural Network
SVM	-	Support Vector Machine
RST	-	Rough Set Theory

CHAPTER 1: INTRODUCTION

1.1. Background

1.1.1. Introduction

Digital transformation has continued to redefine the very dynamics of data management and day-to-day decision making (Waller & Fawcett, 2013). The amount of available data has been rapidly growing and analysing these large data sets is said to become a key basis of competition, underpinning new waves of productivity growth, innovation, risk management practices, and consumer surplus, according to research by MGI and McKinsey's Business Technology Office. This increasing volume and detail of information captured by enterprises coupled with the rise of multimedia, social media, and the Internet of Things will fuel exponential growth in data for the foreseeable future (Manyika et al., 2011).

With this ever growing combination of resources, tools, and applications in big data analytics an opportunity presents itself in enhancing the fight against financial fraud. Financial fraud has been a source of major concern for several organizations and governments alike (Albashrawi, 2016). Forty years ago, banking fraud for example, might have involved simply forging an account holder's signature on a withdrawal slip. Now the speed and intricacy of the schemes are baffling (Corbo, Giovine & Wigley, 2017). Fraudsters continue to grow their schemes executing them with meticulous precision while leaving little to chance.

Financial Institutions across the globe are all susceptible to the problem of fraud but perhaps as prevalent as the problem might be it can be difficult to address. Factors such as: the sheer volume of transactions handled by most institutions versus the relatively small number of fraudulent transactions, the speed with which technology allows fraudsters to operate, poor or incomplete data, and the lack of information sharing among financial institutions are among the contributors to the challenge (Corbo et al., 2017).

1.1.2. Fraud

For a thorough investigation of the subject of fraud a precise and clear definition of the topic is required. Van Vlasselaer, Eliassi-Rad, Akoglu, Snoeck, & Baesens (2016), define fraud as **an uncommon, well-considered, imperceptibly concealed, time-evolving and often carefully organized crime which appears in many types of forms**. This definition given by Baesens et al., (2015) highlights key characteristics of fraud that pose a challenge to the development of fraud-detection systems.

The first characteristic points out the fact that fraud is an *uncommon* phenomenon that is distinct from others. Only a minority of the involved population of cases concern fraud and even more limiting is the fact that only a few of these cases would be known to concern fraud. This not only makes it difficult to detect fraudulent activity but also to learn from historical cases in order to build a powerful fraud-detection system since only few examples are known.

In addition fraudsters try as much as possible to blend in so as not to stand out and get noticed. This makes fraud *imperceptibly concealed,* as fraudsters craft *well considered* schemes to achieve their end goal. Through *careful organized* planning they are able to develop techniques that *evolve in time* along with, or better than fraud-detection mechanisms.

A final element in the description of fraud provided by Van Vlasselaer et al., (2016) points to the *different types of forms* in which fraud occurs. This refers to both the wide set of techniques and approaches used by fraudsters as well as the different settings in which fraud occurs. Examples include: Credit card fraud, Insurance fraud, Corruption, Counterfeit, Money laundering, Telecommunication fraud, Tax evasion etc.

In this fight against fraud, Baesens, Van Vlasselaer, & Verbeke, (2015), lists two components that are essential parts of almost any effective strategy to fight fraud: fraud detection and fraud prevention. Fraud detection involves distinguishing fraudulent financial data from authentic data, thereby disclosing fraudulent behaviour or activities and enabling decision makers to develop appropriate strategies to decrease the impact of fraud (Ngai, Hu, Wong, Chen, & Sun, 2011). Fraud prevention, on the other hand, refers to measures put in place to prevent or reduce fraudulent activity.

1.1.3. Fraud Detection & Machine learning

Fraud detection involves distinguishing fraudulent financial data from authentic data (Ngai et al., 2011). Fundamentally similar is the concept of fraud prevention which refers to measures put in place to prevent or reduce fraudulent activity and both tools should essentially complement each other to reduce fraud (Baesens et al., 2015). The difference between both is clear-cut; the former is an ex post approach whereas the latter an ex-ante approach, however, what is apparent in both is the need to understand and anticipate fraudsters actions.

The classical approach used to detect fraud is an *expert-based approach*[1]. It typically involves a manual investigation of a suspicious case that may be signalled, for instance by a customer complaining of being charged for transactions they did not initiate. Such an occurrence may indicate a new fraud mechanism. Comprehension of this mechanism or pattern allows extending the fraud detection mechanism that is often implemented as a *rule base or engine*[2] that describes the newly detected fraud mechanism. This approach however, suffers from a number of disadvantages. One of such is that fraudster can also learn the business rules that block or expose them through trial and error and find ways to by-pass them. Moreover, since the rules are based on past experience, new emerging fraud patterns are not automatically flagged or signalled (Baesens et al., 2015).

Fraud detection and prevention tools have been evolving over the past decade in order to address the problem and to provide reliable solutions. Machine learning is such a tool that addresses the question of how to build computers that improve automatically through experience (Waltz, 1988). It is one of today's most rapidly growing technical fields, lying at the intersection of computer science and statistics, and at the core of artificial intelligence and data science. Recent progress in machine learning has been driven both by the development of new learning algorithms and theory and by the ongoing explosion in the availability of online data and low-cost computation (Jordan & Mitchell, 2015). It has been a major contributor for detecting different types of financial fraud through its diverse methods, such as, logistic regression, decision trees, Support Vector Machines (SVMs) and naïve Bayes (Ngai et al., 2011).

[1] A system builds on the experience, intuition and business or domain knowledge of the fraud analyst
[2] A set of If-Then rules that trigger an alert or signal when a rule is broken

According to Ngai et al. (2011), an effective fraud-detection and prevention system combines different tools such as - supervised learning, unsupervised learning and semi-supervised learning – which have different possibilities and limitations and therefore reinforce each other when applied in a combined setup.

In supervised methods, models are trained to discriminate between fraudulent and non-fraudulent behaviour, so that new observations can be assigned to classes in order to optimize some measure of classification performance. This requires one to be confident about the true classes of the original data used to build the models; uncertainty is introduced when legitimate transactions are mistakenly reported as fraud or when fraudulent observations are not identified as such. Supervised methods require that one has examples of both classes, and they can only be used to detect frauds of a type that have previously occurred. In addition, these methods also suffer from the problem of unbalanced class sizes: in fraud detection problems, the legitimate transactions generally far outnumber the fraudulent ones and this imbalance can cause misspecification of models (Bolton, Hand et al., 2001).

In contrast, unsupervised methods simply seek those accounts, customers, etc. whose behaviour is 'unusual' (Bolton et al., 2001). One would model a baseline distribution that represents normal behaviour and then attempt to detect observations that show greatest departure from this norm. These can then be examined more closely. Outliers are a basic form of nonstandard observation that can be used for fraud detection. It is particularly useful to notice that unsupervised learning techniques can detect both old and new fraud types since they are not bounded to the fraud patterns that are encapsulated in the labelled training samples like supervised learning techniques do.

The third major emerging machine-learning paradigm is reinforcement learning. In this method, the information available in the training data is intermediate between supervised and unsupervised learning (Horvitz & Mulligan, 2015). Instead of training examples that indicate the correct output for a given input, the training data in reinforcement learning are assumed to provide only an indication as to whether an action is correct or not; if an action is incorrect, there remains the problem of finding the correct action. Although simplified versions of reinforcement learning known as *bandit problems*[3] are studied, where it is assumed that rewards are provided after each

[3] Bandit problems are problems in the area of sequential selection of experiments, and they are related to stopping rule problems through the theorem of Gittins and Jones (1974).

action, reinforcement learning problems typically involve a general control-theoretic setting in which the learning task is to learn a control strategy (a "policy") for an agent acting in an unknown dynamical environment, where that learned strategy is trained to choose actions for any given state, with the objective of maximizing its expected reward over time (Commission et. al., 2013). The ties to research in control theory and operations research have increased over the years, with formulations such as Markov decision processes and partially observed Markov decision processes providing points of contact. (Jordan & Mitchell, 2015).

1.2. Problem Statement

In order to have efficient operations that minimize risk it is key to have high quality, efficient and effective business processes. Business processes should always add value to customers and mitigate risks. While many institutions blame technology or governance as the cause of fraud, many cases of major internal fraud can be traced back to inadequate (or non-existent) business processes that allowed fraudsters to abuse the service. Fraud can have a large impact on the reputation of an institution, and the industry as a whole. In the recent past for example, large cases of fraud in mobile money were reported to have occurred causing financial damages of millions of dollars. The process of preventing fraud includes conducting assessments to understand where fraud could be detected and prevented and establishing effective controls (IFC & MasterCard Foundation, 2016).

The modern approach to fraud detection is to use statistical methods such as machine learning. For example Neural Networks (NN) are used in industrial products by Visa and MasterCard (Patidar, Sharma, & others, 2011) and not forgetting M-Pesa (Purcell, 2016). There are numerous machine learning techniques in use today but regardless of the method chosen the objective is to show comparable (or better) recognition performance in terms of the precision and recall metrics.

This study provides a comparative analysis of the use of NNs and SVMs. SVMs have been noted in recent years to show superior performance across different applications. By implementing this techniques, the intention is to raise detection rates of known fraudulent activity and decrease the false positive rate for unknown fraud cases. In contrast to the more widely used Neural Networks which are prone to local minima, overfitting and noise (Meyer, Leisch, & Hornik, 2003), the study sets out to compare it to SVMs which can obtain global solutions with good generalization error.

1.3. Research Objectives

i. To construct a fraud detection model that utilizes NNs
ii. To construct a fraud detection model that utilizes SVMs
iii. To provide a comparative analysis on the performance of NNs and SVMs

1.4. Research Questions

i. What is the performance of NNs in fraud detection?
ii. What is the performance of SVMs in fraud detection?
iii. What is the difference in performance between NNs and SVMs methods?

1.5. Significance of study

Financial loss due to fraud bears economic, reputational and legal consequences for any organization. In most countries, financial platforms are required to demonstrate rigorous measures to guard against money loses.

Institutions providing financial services stand to benefit most from this study. Due to growing regulations in most countries, it is compulsory for institutions to report fraudulent activities. Therefore, fraud detection is vitally important for institutions to be able to effectively run financial services and prevent reputation risks because of the stigma associated with fraud. A successful fraud detection system or strategy is seen as an important advantage in the financial industry and has led to growing investment and research into the matter globally from academia, industry and government (Baesens et al., 2015). Despite this however, the results are seldom published in the public domain which in turn only serves to hamper overall progress in Financial Fraud research.

This paper addresses some of the problems associated with detecting fraud, and publishes comparative results on the performance of machine learning methods based on NNs and SVMs. This will aid in further research in the field of fraud detection and machine learning. It is interesting to note that, in most cases, researchers claim that SVMs match or outperform NNs in classification problems (Patidar et al., 2011). This claims shall be put to the test.

CHAPTER 2: LITERATURE REVIEW

2.1. Fraud Detection Techniques

Many techniques have been investigated for fraud detection, mainly from the statistical and Machine Learning (Sudjianto et al., 2010) discipline. The easiest technique is to use thresholds on transaction amounts or any other statistical value, such as transaction or expenditure rate (Bolton & Hand, 2002). Among the Machine Learning techniques, mostly used are NNs, SVMs, Bayesian network models, and naïve Bayes scoring. In this section the use of NN and SVM Machine Learning methods is discussed in detail while examining the results in their application.

2.1.1. Neural Networks

Neural networks are widely used in fraud detection, particularly in credit cards. They are the classifiers underlying commercial systems such as VISA which implements neural networks in the fraud detection tool RST performing real-time scoring of transactions (Patidar et al., 2011). The M-PESA Mobile Money service has deployed Minotaur™ Fraud Management Solution based on the use of business rules and neural networks[4]. Typically, feed-forward networks with only three layers (input, hidden, and output layers) are used in fraud detection. The input to the neural network is the vector of features. The signal emitted by the output unit is the probability of the activity being criminal, which is used as a suspicion score. Given enough hidden units and proper nonlinearities and weights, three-layer neural nets are able to implement a universal function approximator (Haykin, 2009). Backpropagation is commonly used for training. The weights are initialized with random values, which are then changed in the direction that minimizes training error. More complex setups with two hidden layers, or strategies other than backpropagation are possible, but uncommon. Looking also at its application across various fields, Husain & Vohra (2017) give an overview of existing applications of machine learning in the financial sector and include a discussion of how Big Data technologies can be applied in the finance sector and traces some leading application paths.

With developments in this field such as recurrent, feedforward, and convolutional NNs, Artificial Neural Networks (ANNs) are gaining in popularity again, and at the same time winning many

[4] "Minotaur™ Fraud Detection Software,"

prizes in recent pattern recognition contests. Because the advanced versions of NNs require even more processing power, they are implemented commonly on graphics processing units (Buczak & Guven, 2016).

Neural networks, however, suffer from some drawbacks. One major issue is the need to select and adjust the structure of the network. The choice of the number of hidden states must be made to optimize learning and generalization. Further, the performance of the classifier is very sensitive to the vector of features chosen, so significant attribute selection and pre-processing (e.g., normalization) are necessary.

Cannady (1998), presented a misuse detection system which attempted to identify instances of network attacks by comparing current activity against the expected actions of an intruder. Using NNs he classified each packet-level data separately. Lippmann et al. (2000), used as similar method utilizing Transmission Control Protocol (TCP)/Internet Protocol (IP) data from the DARPA 1999 challenge where the data set consisted of network packet-level data. Unlike the previous study by Cannady (1998), this system used time windows to perform the detection and classified a group of packets. Thus, the system was able to detect attack types of longer duration because the input was low-level network packet data (as opposed to NetFlow data). However, model over specification meant the granularity was high and they produced predictions still correspond to short durations.

Bivens et al. (2002), incorporate unsupervised learning in their method and describe a complete Intrusion Detection System that employs a pre-processing stage, clustering the normal traffic, normalization, an NN training stage, and an NN decision stage. The first stage used a SOFM, which is a type of unsupervised NN, to learn the normal traffic patterns over time, such as commonly used TCP/IP port numbers. In this manner, the first stage quantized the input features into bins, which were then fed to the second stage, a Multilayer Perceptron (MLP) NN. The MLP network parameters, such as the number of nodes and layers, were determined by the first stage SOFM. Once the MLP training was completed, it started predicting the intrusions. The system can be restarted for a new SOFM to learn a new traffic pattern and a new MLP attack classifier to be trained. This method reported successfully predicting 100% of the normal behaviour. This overall approach was promising, however falls short when examined in totality. What we see is that some attacks were not fully predicted and the FAR for some attacks reached a high of 76%.

Sudjianto et al. (2010), points out another shortfall of NNs. Training neural networks is time consuming for large training datasets, especially if the model is intended to be retrained very often. In addition, backpropagation-trained multilayer perceptrons are prone to overfitting although a number of algorithms exist that address this problem but at the expense of adding complexity to the training process. Finally, neural networks are often treated as "black boxes," and their results can be difficult to interpret.

2.1.2. Support Vector Machines

An SVM maps the input vector into a higher dimensional space. It is a binary classification technique that classifies input instances into two classes. Only the Support Vectors determine the optimal separating hyper-plane to classify input instance into one of the two classes. Support Vectors are the points closest to the separating hyper-plane. During classification, mapped input vectors placed on one side of the separating hyper-plane in the feature space fall into one class while that placed on the other side of the plane fall into the other class. In case the data points are not linearly separable, SVM uses suitable kernel function to map them into higher dimensional space, so that, in that higher dimensional space they become separable (Sharma et al., 2016).

Boser, Guyon, & Vapnik (1992), first discussed the theory and practicalities behind the optimal margin classifier for separable training data - a discussion which was extended to the non-separable case by Cortes & Vapnik (1995). Burges (1998), later gave a detailed and complete tutorial on support vector machines when used for pattern recognition, and a more in depth account of the theory involved. The paper by Lee, Lin, & Wahba (2001), then tackled the task of extending SVMs to multi-category problems, i.e. problems where the data set can be divided into more than two classes. Not long after the work by Keerthi & Lin (2003) emerged useful in investigating what happens when an SVM displays tendencies towards asymptotic behaviour, for example when the separating hyper planes take on very large or small values. This helps in understanding how to prevent an SVM from overfitting or under fitting the training data. Perhaps more practical was Hsu, Chang & Lin (2003), who gave important information on successfully using SVMs in practice. They went further to discuss the cross-validation technique and proposed a grid-search method for obtaining the most appropriate kernel parameters for a new classification problem. In 2004, Kroon & Omlin went ahead to provide an overview of SVM theory and application, including discussions on the kernel trick used to generalise the SVM to high-dimensional feature

space in an attempt to transform non-separable data into separable data. Their paper on SVM application is also useful in getting insight into using the popular libSVM toolkit (Chang & Lin, 2012). All this is highlighted in the study done by Wiese & Omlin (2009), which gives an overview of SVMs.

The Basic Security Module in Lippmann et al. (2000), which showed portions from the DARPA 1998 data set were used to pre-process training and testing data. The study yielded good classification performance in the presence of noise (such as some mislabelling of the training data set) and reported 75% accuracy with no false alarms and 100% accuracy with a 3% FAR. Hu, Liao, & Vemuri (2003), used the Robust SVM, a variation of the SVM where the discriminating hyperplane is averaged to be smoother and the regularization parameter is automatically determined, as the anomaly classifier in their study.

Most intriguing however, is the approach described by Shon & Moon (2007), as a framework for the detection of novel attacks in network traffic. Their approach is a combination of the SOFM, Genetic Algorithm, and SVM. Data was classified using the Enhanced SVM, which is derived from a one-class SVM, and the supervised soft-margin SVM. The first provides the unlabelled classification capability of the one-class SVM, and the second provides the high detection performance of the supervised SVM. The study used the 1999 DARPA intrusion detection data set. To make the data set more realistic, the subset chosen consisted of 1% to 1.5% attacks and 98.5% to 99% normal traffic. The results from the Enhanced SVMs had 87.74% accuracy, a 10.20% FP rate, and a 27.27% FN rate. Those results were substantially better than those from the one-class SVMs, but not as good as soft-margin SVMs. However, the Enhanced SVM can detect novel attack patterns, whereas a soft-margin SVM cannot.

These studies are summarized in greater detail in the survey on data mining and machine learning techniques by Buczak & Guven (2016), which goes into matters to do with the application of SVMs.

2.2. Theoretical Framework

2.2.1. Neural Networks

The original biological motivation for NNs stems from McCulloch & Pitts (1943), who published a seminal model of a neuron as a binary thresholding device in discrete time. It is from this that it is said NNs are inspired by the brain and composed of interconnected artificial neurons capable of certain computations on their inputs (Hornik, Stinchcombe, & White, 1989). The input data activate the neurons in the first layer of the network whose output is the input to the second layer of neurons in the network. Similarly, each layer passes its output to the next layer and the last layer outputs the result. Layers in between the input and output layers are referred to as hidden layers. When an NN is used as a classifier, the output layer generates the final classification category (Buczak & Guven, 2016).

NN classifiers are based on the perceptron by Rosenblatt (1958) and were very popular until the 1990s when SVMs were invented by Boser et al. (1992). Compared to the convex quadratic optimization applied in an SVM, NNs often suffer from local minima and thus long runtimes during learning. Unlike an SVM, as the number of features in an NN increase, it's learning runtime increases. With one or more hidden layers, the NN is able to generate nonlinear models (Buczak & Guven, 2016).

In its simplest form, the perceptron is a structure consisting of three layers: an input layer for presenting input to the perceptron, an association unit layer that acts as a feature detector, and an output response layer. Its task is to calculate an output by applying a threshold to the weighted sum of its inputs. To explain further, it takes a real-valued input vector, calculates a linear combination of these inputs and then gives a 1 if the result is greater than some threshold, or -1 otherwise (Mitchell, 1997). The figure below illustrates the general structure of a perceptron.

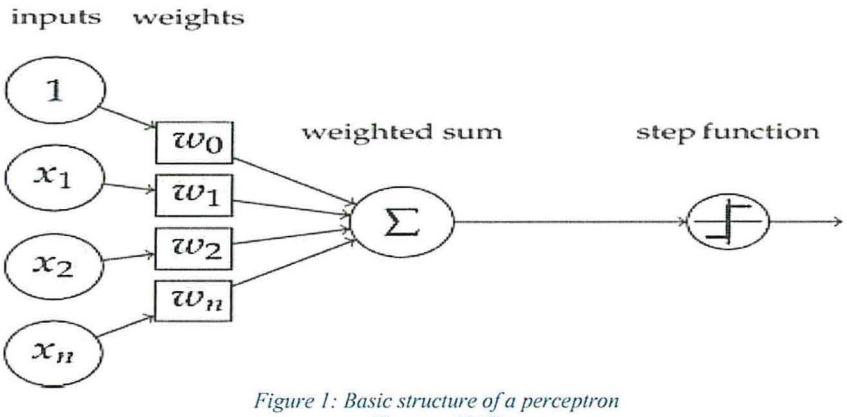

Figure 1: Basic structure of a perceptron (Bargen, 2013)

In the above figure the symbols x_1 to x_n represent the components of a input vector, while w1 to wn represent the weights associated with each of the n dimensions of the input vector. Weight w_0 is associated with a constant input (x_0) of value 1. The quantity ($-w_0$) is therefore a bias which the weighted sum of the input vector components must exceed in order for the perceptron to achieve activation. A perceptron's output is therefore computed using the following equation:

$$0 = \begin{cases} -1 & w_1x_1 + w_2x_2 + \cdots + w_nx_n > -w_0 \\ +1 & otherwise \end{cases} \quad \ldots. (1)$$

To train the perceptron small random values are used to first initialize the weights before training begins. The weights are then updated using the perceptron training rule (Mitchell, 1997) which updates weight w_i associated with input x_i according to the rules below:

$$w_i \leftarrow w_i + \Delta w_i \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

$$\Delta w_i = \eta(d - o)x_i \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3)$$

In equation (3) the symbol d represents the target class or desired output of the training example while o represents the output generated by the perceptron after feeding it with the training example. The learning rate is represented by η and is normally set to a small positive value which can be forced to decay as training progresses in order to limit the effects of local minima on training. The shortcoming of the perceptron is that it can only be used for classification problems with linearly separable data.

Multi-layer networks are formed by organizing multiple perceptrons into several layers to form a forward connected network and the gradient descent algorithm is commonly used in training.

2.2.2. Support Vector Machines

SVMs statistical learning techniques, with strong theoretical foundation have displayed successful application in a range of problems (Cristianini & Shawe-Taylor, 2000). They are closely related to neural networks, and through use of kernel functions, can be considered an alternate way to obtain neural network classifiers.

Rather than minimizing empirical error on training data, SVMs seek to minimize an upper bound on the generalization error. As compared with techniques like neural networks which are prone to local minima, overfitting and noise, SVMs can obtain global solutions with good generalization error. They are more convenient in application, with model selection built into the optimization procedure, and have also been found to outperform neural networks in classification problems (Meyer et al., 2003). Appropriate parameter selection is, however, important to obtain good results with SVM.

During the supervised training process of a classifier training vectors or patterns of the form (x, y), where x is a set of input parameters or features and y denotes class membership, are repeatedly presented to the classifier in an attempt to learn a decision function $D(x)$, which can later be used to make classification decisions on previously unseen data. In the case of the optimal margin training algorithm, these decision functions have to be linear in their parameters, but are not restricted to linear dependencies in their input components x, and can also be expressed in either direct or dual space (Boser et al., 1992).

In its simplest form, the SVM algorithm will construct a hyperplane that completely separates (at least in the linear separable case) the data in such a way that $wx + b > 0$ for all points belonging to one class, and $wx + b < 0$ for all points in the other class (Kroon & Omlin, 2004). In other words, for $D(x) > 0$ pattern x belongs to class A and for $D(x) < 0$ pattern x belongs to class B. In this context, $D(x)$ is referred to as the decision surface or separating hyperplane and all points x which lie on this decision surface satisfy the equation $wx + b = 0$.
Bhattacharyya, Jha, Tharakunnel, & Westland (2011), highlight that the strength of SVMs come from two important properties they possess — kernel representation and margin optimization:

In SVMs, mapping to a high-dimensional feature space and learning the classification task in that space without any additional computational complexity are achieved by the use of a kernel

function. The reason behind the transformation into a higher-dimensional feature is based on Cover's theorem, which states that in cases where data is inseparable in a low-dimensional space, transformation of the data to a higher-dimensional feature space often yields linear separability (Kroon & Omlin, 2004). Schölkopf & Smola (2002), explains that a kernel function can represent the dot product of projections of two data points in a high-dimensional feature space. The high-dimensional space used depends on the selection of a specific kernel function. The classification function used in SVMs can be written in terms of the dot products of the input data points. Thus, using a kernel function, the classification function can be expressed in terms of dot products of projections of input data points in a high-dimensional feature space. With kernel functions, no explicit mapping of data points to the higher-dimensional space happens while they give the SVMs the advantage of learning the classification task in that higher-dimensional space.

The second property of SVMs is the way the best classification function is arrived at. SVMs minimize the risk of overfitting the training data by determining the classification function (a hyper-plane) with maximal margin of separation between the two classes. This property provides SVMs very powerful generalization capability in classification.

In SVMs, the classification function is a hyper-plane separating the different classes of data.

$$(w, x) + b = 0$$

The notation (w, x) represents the dot product of the coefficient vector w and the vector variable x. The solution to a classification problem is then specified by the coefficient vector w. It can be shown that w is a linear combination of data points $xi \cdot i = 1,2,\cdots, m \ i.e. w = \sum a_i x_i, a_i \geq 0$. The data points xi with non-zero αi are called the support vectors. A kernel function k can be defined as $k(x_1, x_1) = ((\Phi x_1), (\Phi x_2))$ *where* $\Phi: X \rightarrow H$ is a mapping of points in the input space X into a higher-dimensional space H. As can be seen, the kernel function implicitly maps the input data points into a higher-dimensional space and return the dot product without actually performing the mapping or computing the dot product. There are several kernel functions suggested for SVMs. Some of the widely used kernel functions include, linear function, $k(x_1, x_2) = (x_1, x_2)$; Gaussian radial basis function (RBF) , $k(x_1, x_1) = e^{-\sigma \|x_1 - x_2\|^2}$ and polynomial function , $k(x_1, x_2) = (x_1, x_2)^d$. The selection of a specific kernel function for an application depends on the nature of the classification task and the input data set. As can be inferred, the performance of SVMs is greatly depended on the specific kernel function used (Bhattacharyya et al., 2011).

2.3. Conceptual Framework

Over the years, along with the evolution of fraud detection methods, perpetrators of fraud have also been evolving their fraud practices to avoid detection (Bolton et al., 2001). Therefore, Mobile Money fraud detection methods need constant innovation. In this study, we evaluate two advanced Machine Learning approaches, Support Vector Machines and Neural Networks, as part of an attempt to better detect Mobile Money fraud.

Underlying the fraud detection model are the basic and derived feature. The basic features are obtained from the characteristics of the particular transaction carried out by the customer e.g. transaction amount, transaction time etc. The derived features on the other hand incorporate information that can be found from the transaction details, historical entries as well as the customer profile. Examples of these features would include the transaction amount carried out over an n-period, average income received by the customer over an n-period etc. These features form the foundation upon which predictive analysis can begin.

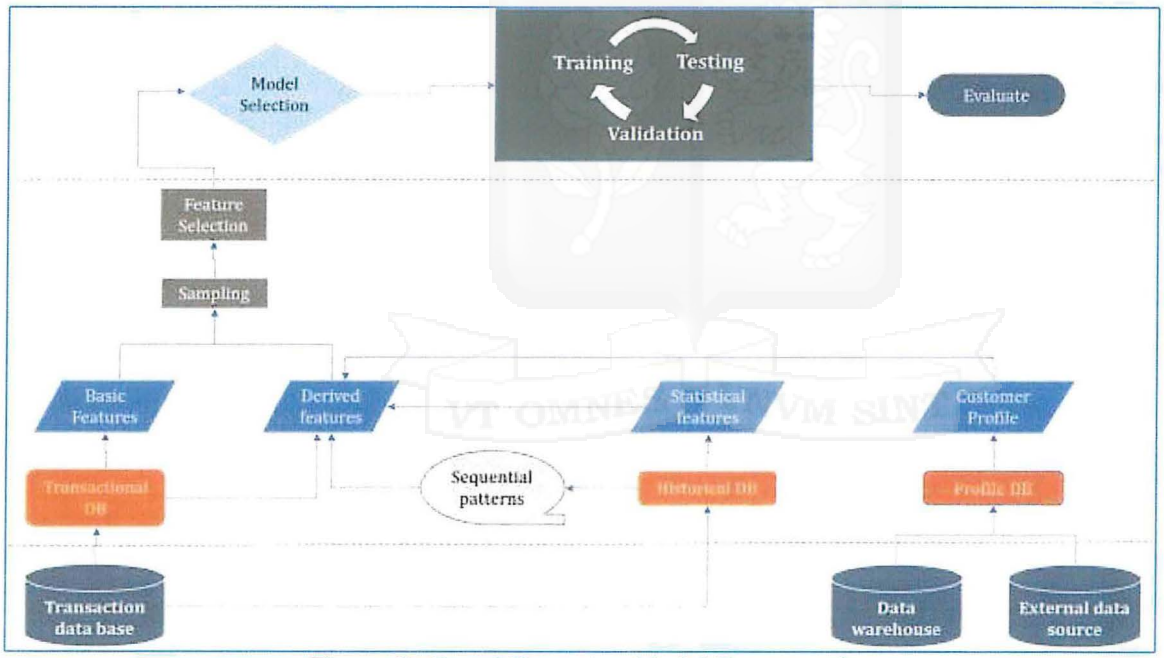

Figure 2: Fraud Detection Model Framework

After features are derived sampling is conducted to identify the data that will be used to train and test the model. Before this is done, however, feature selection is required in order to determine the suitability of the features derived earlier. This is perhaps the most important stage in the model framework as the features could either act as a deterrent or enhancer to the model.

After model selection pre-processing of the data is required in order to have appropriate values for model training.

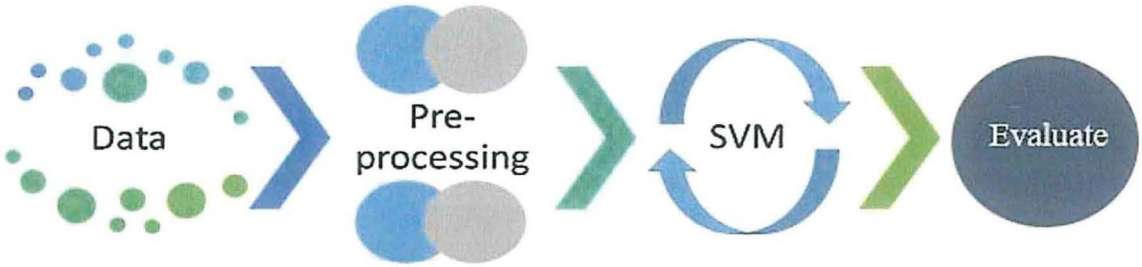

Figure 3: SVM Conceptual Model

Figure 4: NN Conceptual Model

This study aims at providing a comparative analysis on the use of Neural Networks and Support Vector Machines. The choice to focus on SVMs as opposed to other Machine Learning methods is influenced by existing literature that claims that SVMs are already known as one of the best learning algorithm for binary classification.

A series of experiments will be run on the data set using NNs and SVMs and the resulting performances of each algorithm will be computed. These results will then be analytically compared to see how the two algorithms compare to each other when applied to a data set.

CHAPTER 3: METHODOLOGY

3.0. Introduction

This chapter presents the step by step research methodology used in achieving the earlier laid out objectives.

3.1. Research Design

This study takes an applied exploratory approach to investigate the effectiveness of using SVMs in Fraud detection in comparison to NNs.

3.2. Population and Sampling

The datasets to be used contains transactions made by credit cards in September 2013 by European cardholders. This dataset presents transactions that occurred in two days, where there are 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

It contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, the original features and more background information about the data is withheld. Features V1, V2, ..., V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount, this feature can be used for example-dependant cost-sensitive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

The dataset was collected and analysed during a research collaboration of Worldline and the Machine Learning Group of ULB (Université Libre de Bruxelles) on big data mining and fraud detection. Enabled by: Andrea Dal Pozzolo, Olivier Caelen, Reid A. Johnson and Gianluca Bontempi. Calibrating Probability with Under sampling for Unbalanced Classification. In Symposium on Computational Intelligence and Data Mining (CIDM), IEEE, 2015.

3.3. Data Processing

Both Machine Learning methods will be subjected to three phases: Training, Testing and Validation. Buczak & Guven (2016), point out that Machine Learning and Data Mining methods often have parameters such as the number of layers and nodes for a NN. After the training is complete, there are usually several models (e.g., NNs) available. To decide which one to use and have a good estimation of the error it will achieve on a test set, there should be a third separate data set, the validation data set. The model that performs the best on the validation data should be the model used, and should not be fine-tuned depending on its accuracy on the test data set. Otherwise, the accuracy reported is optimistic and might not reflect the accuracy that would be obtained on another test set similar to but slightly different from the existing test set.

Given the class imbalance ratio, the accuracy measure to be used will be the Area Under the Precision-Recall Curve (AUPRC). This is because confusion matrix accuracy is not meaningful for unbalanced classification.

3.4. Data Analysis

3.4.1. Soft-Margin SVM

Borrowing from Shon & Moon (2007), they describe a soft margin SVM learning algorithm written by Cortes & Vapnik, (1995), which is sometimes called c-SVM. This SVM classifier has a slack variable and penalty function for solving non-separable problems. First, given a set of points $x_i \in R^d, i = 1, \ldots, l$ and each point x_i belongs to either of two classes with the label $y_i \in \{-1,1\}$. These two classes can be applied to anomaly detection with the positive class representing normal and negative class representing abnormal. Suppose there exists a hyper-plane $w^T x_i + b = 0$ that separates the positive examples from the negative examples. That is, all the training examples satisfy:

$$w^T x_i + b \geq +1 \; for \; all \; x_i \in P \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4)$$

$$w^T x_i + b \leq -1 \; for \; all \; x_i \in N \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (5)$$

w is an adjustable weight vector, x_i is the input vector and b is the bias term.

Equivalently:

$$y_i(w^T x_i + b) \geq 1 \; for \; all \; i = 1, \ldots, N \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (6)$$

In this case, we say the set is linearly separable.

In Fig. 5, the distance between the hyper-plane and $f(x)$ is $\frac{1}{||w||}$. The margin of the separating hyper-plane is defined to be $\frac{2}{||w||}$. Hence, the learning problem is reformulated as minimizing $||w||^2 = w^T w$ is subject to the constraints of linear separation. This is equivalent to maximizing the hyper-plane distance between the two classes, of which the maximum distance is called the support vector. The optimization is now a convex quadratic programming problem.

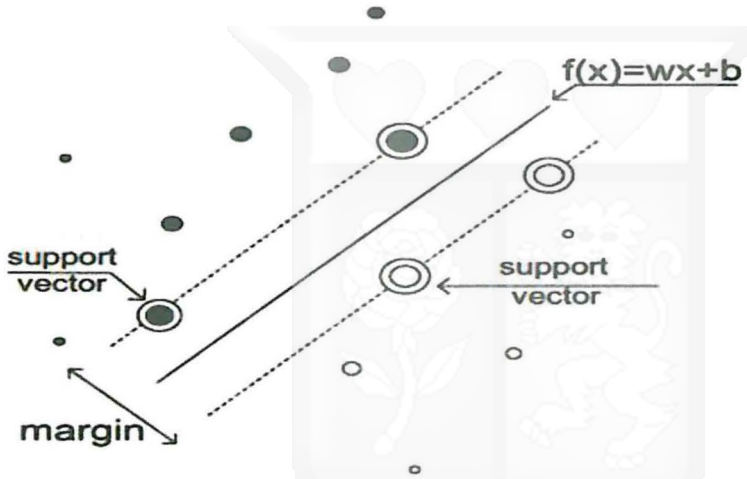

Figure 5: SVM hyper-plane
Source: (Wiese & Omlin, 2009)

$$\text{Minimize}_{\text{w,b}} \quad \phi(w) = \frac{1}{2}||w||^2 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (7)$$

$$\text{Subject to} \quad y_i(w^T x_i + b) \geq 1 \; for \; all \; i = 1, \ldots, l$$

Since $\phi(w) = \frac{1}{2}||w||^2$ is convex in w and the constraints are linear in w and b, we can be sure that this problem has a global optimum solution. This has the advantage that parameters in quadratic programming affect only the training time, and not the quality of the solution. This problem is tractable, but anomalies in fraud detection show characteristics of non-linearity, and as a result they are more difficult to classify. In order to proceed to such non-separable and non-linear cases, it is useful to consider the dual problem as outlined below:

The Lagrange for this problem is

$$L(w, b, \Lambda) = \frac{1}{2}||w||^2 - \sum_{i=1}^{l} \lambda_i [y_i(w^T x_i + b) - 1] \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (8)$$

Where $\Lambda = (\lambda_1, \ldots, \lambda_l)^T$ are the Lagrange multipliers, one for each data point. The solution to this quadratic programming problem is given by maximizing L with respect to $\Lambda \geq 0$ and minimizing $\frac{1}{2}||w||^2$ with respect to w and b. Note that the Lagrange multipliers are only non-zero when $y_i(w^T x_i + b) = 1$, and the vectors for this case are called support vectors, since they lie closest to the separating hyper-plane. However, in the non-separable case, forcing zero training error leads to poor generalization. We introduce the soft margin SVM using a vector of slack variables $\mathfrak{Z} = (\psi_1, \ldots, \psi_l)^T$ that measure the amount of violation of the constraints, taking into account the fact that some data points may be misclassified.

The equation is now:

$$\text{Minimize}_{w,b,\mathfrak{Z}} \qquad \phi(w, b, \mathfrak{Z}) = \frac{1}{2}||w||^2 + C \sum_{i=1}^{l} \psi_i^k \quad \ldots\ldots\ldots\ldots\ldots\ldots (9)$$

$$\text{Subject to} \qquad y_i(w^T \phi(x_i) + b) \geq 1 - \psi_i, \psi_i \geq 0, \quad i = 1, \ldots, l \quad \ldots\ldots\ldots\ldots (10)$$

where C is a regularization parameter that controls the trade-off between maximizing the margin and minimizing the training error. The effects of C are crucial. If C is too small, insufficient stress is placed on fitting the training data. If C is too large, the algorithm will over fit the dataset.

In practice, a typical SVM approach, such as the soft margin SVM, showed excellent performance more often than other machine learning methods (Akande, Owolabi, Twaha, & Olatunji, 2014). For intrusion detection applications, supervised machine learning approaches based on SVM methods proved superior to intrusion detection approaches using artificial neural networks. Therefore, the high classification capability and processing performance of the soft margin SVM approach is useful for anomaly detection. However, because the soft margin SVM is a supervised learning approach, it requires datasets to be labelled.

3.4.2. Feedforward Neural Network

The neural network that will be used in this fraud detection framework is a multi-layer, feed-forward network that uses two training passes through the data set and borrows from the work by Patidar et al., (2011). The first training pass involves a process of prototype cell commitment in which exemplars from the training set are stored in the weights between the first and second (middle) layer cells of the network. A final training pass determines local a posteriori probabilities associated with each of these prototype cells. The objective of the neural network training process is to arrive at a trained network that produces a fraud score that gives the best ranking of the Mobile Money transactions. If the ranking were perfect, all of the high scoring transactions down to some threshold would be fraud; below this threshold, only good transactions would be ranked. However, perfect separation of frauds from goods is not possible due to the inherently non-separable nature of the fraud and good distributions in the selected pattern recognition space.

The back propagation learning rule is a standard learning technique. It performs a gradient descent in the error/ weights space. To improve the efficiency, a momentum term is introduced, which moves the correction of the weights in the direction compliant with the last weight correction. It is a multi-layer feed forward network that is trained by supervised learning. A standard back propagation network consists of 3 layers, an input, an output and a hidden layer. The processing elements of both input and output layer are fully connected with the processing elements of the hidden layer. The fact that it is feed forward means that there are no recurrent loops in the network. In standard back propagation this can never happen because the input for each processing element always comes from the previous layer (except the input layer, of course). The output of a node never returns at the same node, because cycles are not allowed in the network. This, again, is a large simplification compared with the real brain because the brain itself appears to contain many recurrent loops.

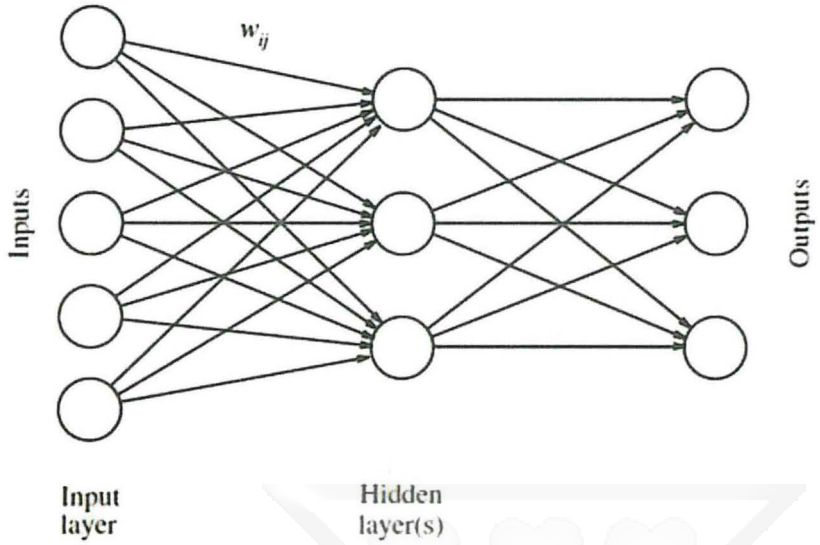

Figure 6: FFNN Framework
Source: (Ripley, 2002)

The feed-forward neural network is defined by:

$$O_k(x) \equiv Z_k = g\left(\sum_{j=1}^{n_H} w_{kj}^{(1)} f\left(\sum_{i=1}^{d} w_{ji}^{(0)} x_i + w_{jo}^{(0)}\right) + w_{k0}^{(1)}\right) \ldots\ldots\ldots (11)$$

where x_i are the d inputs, w_{kj} are the weights between the output y and the hidden layer, w_{ji} are the weights from the ith input x_i to the jth neuron of the hidden layer, n_H is the number of neurons in the hidden layer, k is the number of output neurons and $Z_k \equiv O_k(x)$ are the k outputs. $f(\ldots)$ is a non-linear transfer function (e.g. $f(\tanh(x))$. Linear outputs $g(\ldots)$, will be used. Super indices in parenthesis are used to explicitly state each element's position in the network.

The size of the hidden layer is a fundamental question often raised in the application of multilayer FFNN to real world problems. It is usually determined experimentally and by empirical guidelines. For a network of a reasonable size, the size of hidden nodes needs to be only a fraction of the input layer (Lin et al., 1996). A simple rule of thumb is:

$$h = \frac{m+n}{2} + \{\ldots 0 \ldots 1 \ldots 2 \ldots\} \ldots\ldots\ldots\ldots\ldots\ldots (12)$$

where, the symbol, h is the number of neurons in the hidden layer, m is the number of neurons in the input layer and n is the number of neurons in the output layer. The $\{\ldots 0 \ldots 1 \ldots 2 \ldots\}$ part of the equation means that if a network fails to converge to a solution with the result of the integer

division, it may be that more hidden neurons are needed. So, their number is incremented until convergence (Hilas & Mastorocostas, 2008).

Supervised learning means that the network is repeatedly presented with input/output pairs (I,O) provided by a supervisor, where O is the output the network should produce when presented with input I. These input/output pairs specify the activation patterns of the input and output layer. The network has to find an internal representation that results in the wanted input/output behaviour. To achieve this, back propagation uses a two-phase propagate-adapt cycle:

i. *First Phase*: In the first phase the input is presented to the network and the activation of each of the nodes (processing elements) of the input layer is propagated to the hidden layer, where each node sums its input and propagates its calculated output to the next layer add a constant (the 'bias') and take a fixed function φ_h of the result. The output units are of the same form, but with output function φ_o. Thus:

$$y_k = \varphi_o(\propto_k + \sum_h w_{hk}\varphi_h\ (\propto_h + \sum_i w_{ih}x_i)) \ldots\ldots\ldots (13)$$

The nodes in the output layer calculate their activations in the same way as the nodes in the hidden layer. The 'activation function' φh of the hidden layer units is almost always taken to be the logistic function:

$$l(z) = \frac{e^z}{1+e^z} \ldots\ldots\ldots\ldots\ldots (14)$$

ii. *Second Phase*: In the second phase, the output of the network is compared with the desired output given by the supervisor and for each output node the error is calculated. Then the error signals are transmitted to the hidden layer where for each node its contribution to the total error is calculated. Based on the error signals received, connection weights are then adapted by each node to cause the network to converge toward a state that allows all the training patterns (input/output pairs) to be encoded.

3.5. Performance Measure

In lieu of measuring accuracy, precision metric is frequently adopted in the research community to provide assessments of imbalanced learning problems.

Area Under the Curve (AUC) of the Receiver Operating Characteristic (ROC) has a disadvantage in the case of highly skewed data sets. It is observed that the ROC curve may provide an overly optimistic view of an algorithm's performance. Under such situations, the Precision Recall (PR) curves can provide a more informative representation of performance assessment (Davis & Goadrich, 2006).

Davis & Goadrich (2006), outline that given a confusion matrix and the definition of precision as given by equation (15) and recall by equation (16), the PR curve is defined by plotting precision rate over the recall rate. PR curves exhibit a strong correspondence to ROC curves: A curve dominates in ROC space if and only if it dominates in PR space. However, an algorithm that optimizes the AUC in the ROC space is not guaranteed to optimize the AUC in the PR space. Moreover, while the objective of ROC curves is to be in the upper left hand of the ROC space, a dominant PR curve resides in the upper right hand of the PR space. PR space also characterizes curves analogous to the convex hull in the ROC space, namely, the achievable PR curve. Hence, PR space has all the analogous benefits of ROC space, making it an effective evaluation technique(He & Garcia, 2009).

Performance Measure	*Explanation*
$Precision = \frac{TP}{TP+FP}$ ..... (15) (Positive Prediction Accuracy)	How good a model is at detecting the positives.
$Sensitivity/Recall = \frac{TP}{TP+FN}$ ..... (16) (True Positive Rate)	How many of the positively classified were correctly classified.
$Specificity = \frac{TN}{TN+FP}$..... (17) (True Negative Rate)	How good a model is at avoiding false alarms
$F1 - Score = 2 * \frac{Precision*Recall}{Precision+Recal}$ ....(18)	Measures accuracy by taking the harmonic average of the precision and recall.
$G - measure = \sqrt{Precision * Recall}$ ... (19)	Measures accuracy by taking the geometric average of the precision and recall.

Table 1: Performance Measures

CHAPTER 4: EXPERIMENTAL SETUP

This section describes the process and parameters used in pre-processing before training and testing the models.

4.1. Data

The table below gives a summary of the original data set described earlier in the population and sampling in chapter 3:

No. of observations	284,807
No. of features	30
No. of Fraudulent transactions:	492
No. of Normal transactions are:	284,315
Normal transactions (%)	99.83
Fraudulent transactions (%)	0.17

Table 2: Summary of data set

The correlation between features was calculated and presented in a heat map so as to gain a better understanding of the features given. The figure below gives this output, showing correlation values above 0.8 and below -0.8.

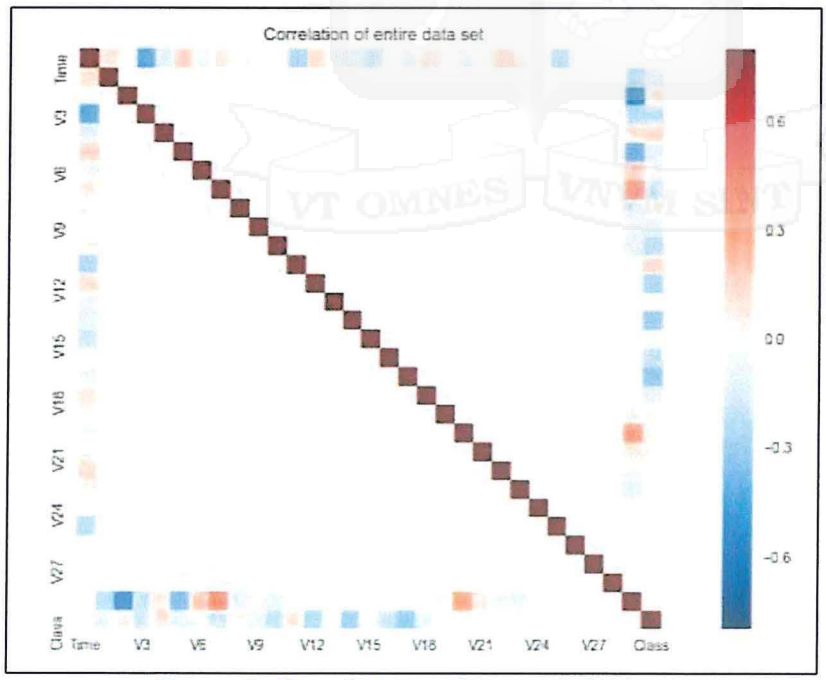

Figure 7: Correlation within entire data set

From the heat map above it is observed that the Class (Fraud or Normal) correlated more with feature within the range of V1 – V18.

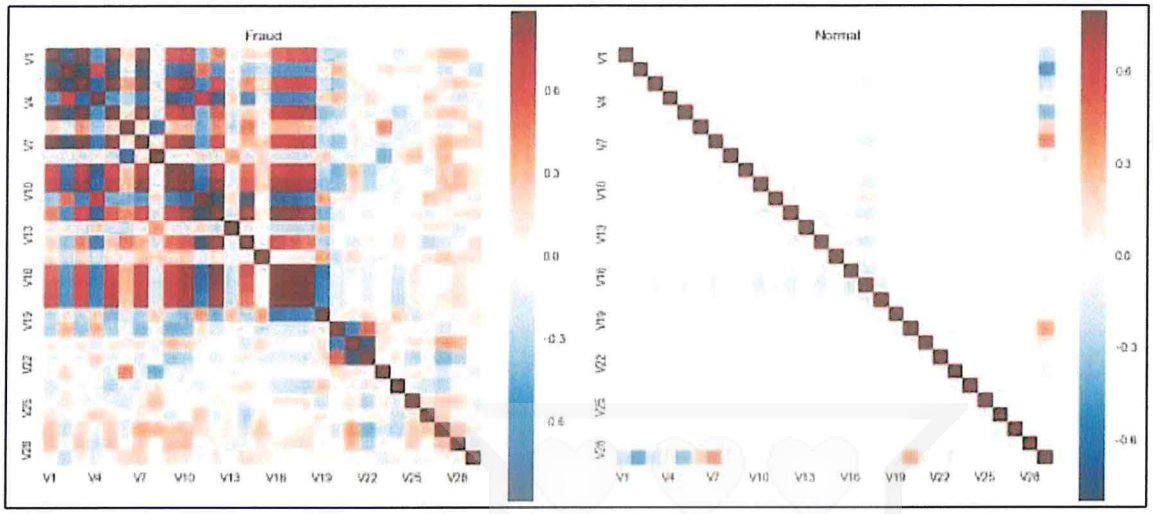

Figure 8: Correlation of features across Class

Similarly after separating both the Fraud and Normal Classes a strong correlations of features between V1-V18 in the Fraud heat map can be observed with slightly less correlation between other features. For the Normal Class there is much less correlation between features.

Delving deeper into the descriptive statistics one notes that the total Fraud amount, over the two day period, was 60,127.97 with the highest individual fraudulent transaction amounting to 2,125.87. Moreover, in the two days the total value of transactions given by the bank was 25,162,590.01 of which the proportion of fraudulent transactions was 0.24%.

4.2. Feature selection

Although the data set provides pre-selected features that have been transformed using PCA further scrutiny of these features is required to determine their suitability for model training.

By plotting the distribution of the features based on the Class one is able to pick out those features that show distinguishable distributions. The features that reveal a difference in the distribution would be suitable for training the model as they provide better classification potential.

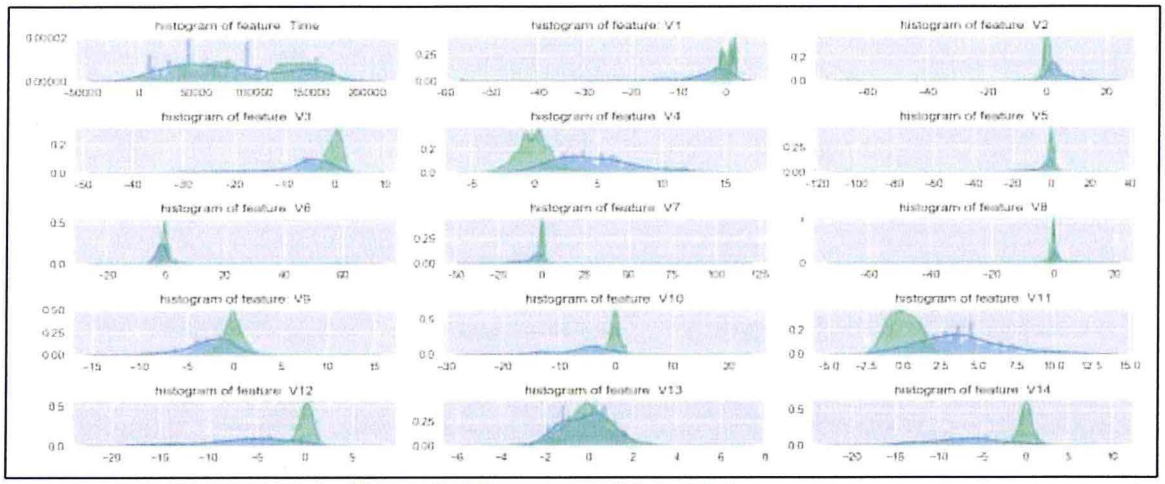

Figure 9: Histogram of first 15 features

The plots above display the frequency distribution of the first half of features in the original data set. The blue region represents Normal transactions whereas the green region represents the Fraud transactions.

Taking the example of feature V13 we see that the distribution of both the Normal and Fraud transaction overlap and are as such almost indistinguishable. Consequently, we drop this feature as it does not present good classification potential. On the contrary, feature V14 clearly shows a distinguishable distribution between the Normal and Fraud cases and as such would provide good classification potential. We thus retain such a feature.

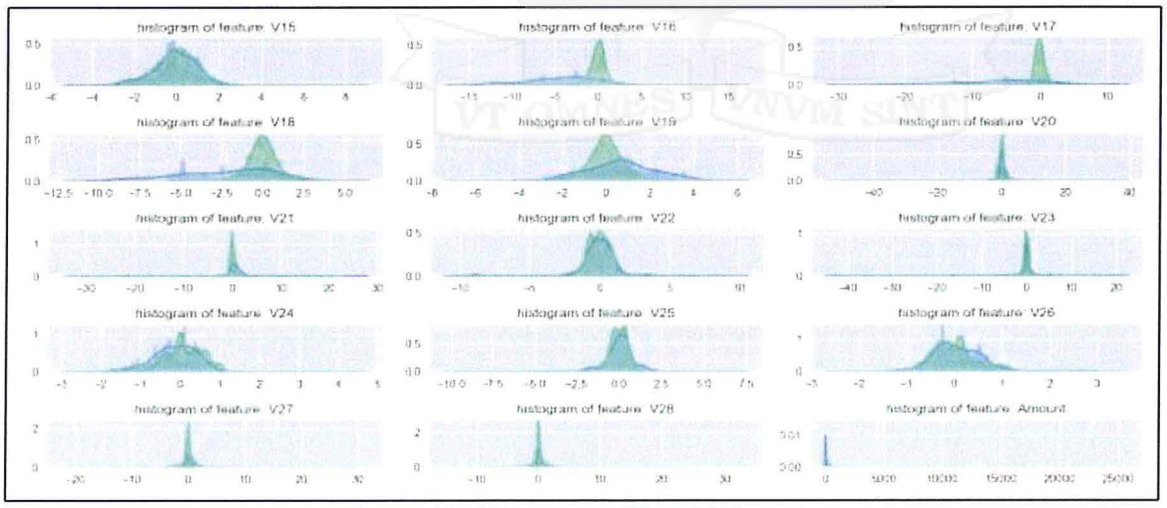

Figure 10: Histogram of last 15 features

This feature selection procedure was also applied to the second half of features in the data set.

The results of the feature selection lead to the eventual removal of the following features with similar distributions across Fraudulent and Normal transactions : 'V28', 'V27', 'V26', 'V25', 'V24', 'V23', 'V22', 'V21', 'V20', 'V15' ,'V13', 'V8', 'V6', 'V5'.

To check whether this feature reduction is useful we visualize our data using t-SNE. PCA is sometimes used to reduce dimensionality but it may not be the best method since it is a linear and parametric method. T-SNE on the other hand is a non-parametric and non-linear method that assess the inherent data structure.

Firstly we employ the original data, then the new data set. The expectation is that the second scatter plot will show a clearer contrast between the normal and the fraudulent transactions. If this is the case, it signals that the work done in the feature reduction stage of the analysis was beneficial to helping the model understand the data.

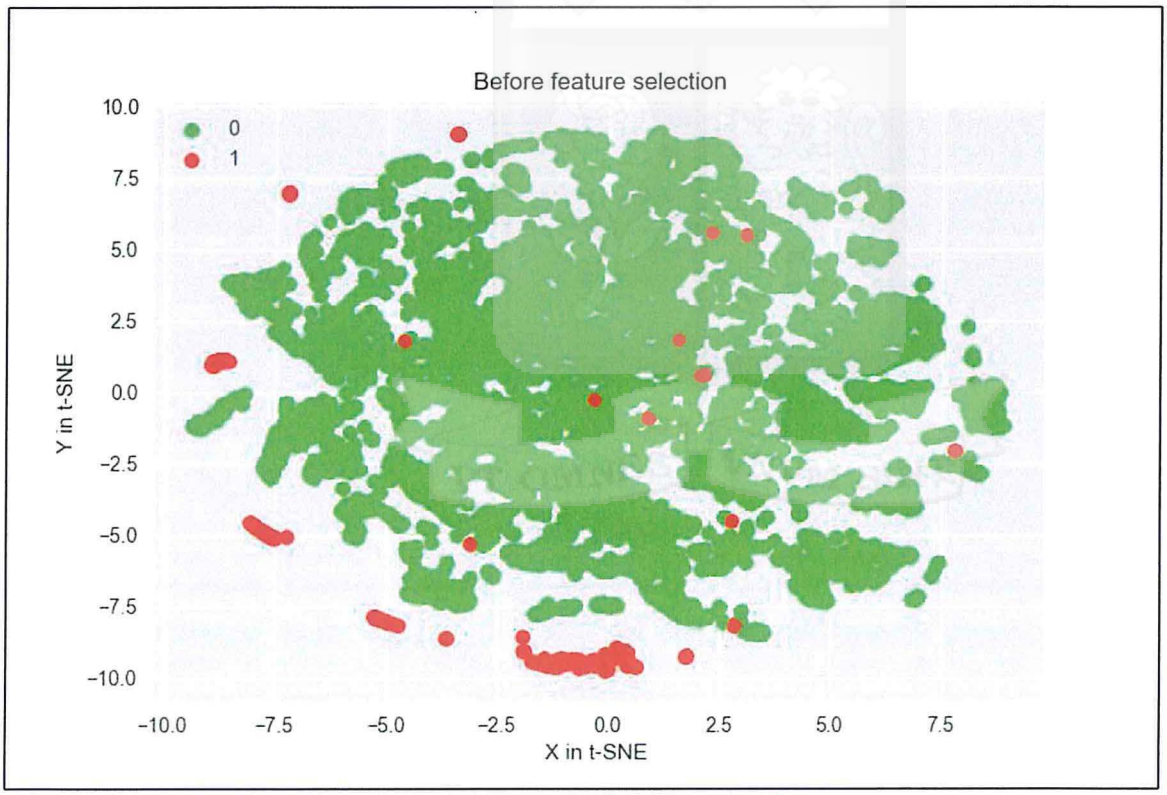

Figure 11: t-SNE plot before feature selection

From the original data set there seems to be a pattern of fraudulent transactions forming on the bottom left region of the plot. The remainder of the fraudulent transactions are scattered within the rest of the data.

Note: Due to processing limitations only 10,000 of the 284,315 normal transactions were used for this visualization.

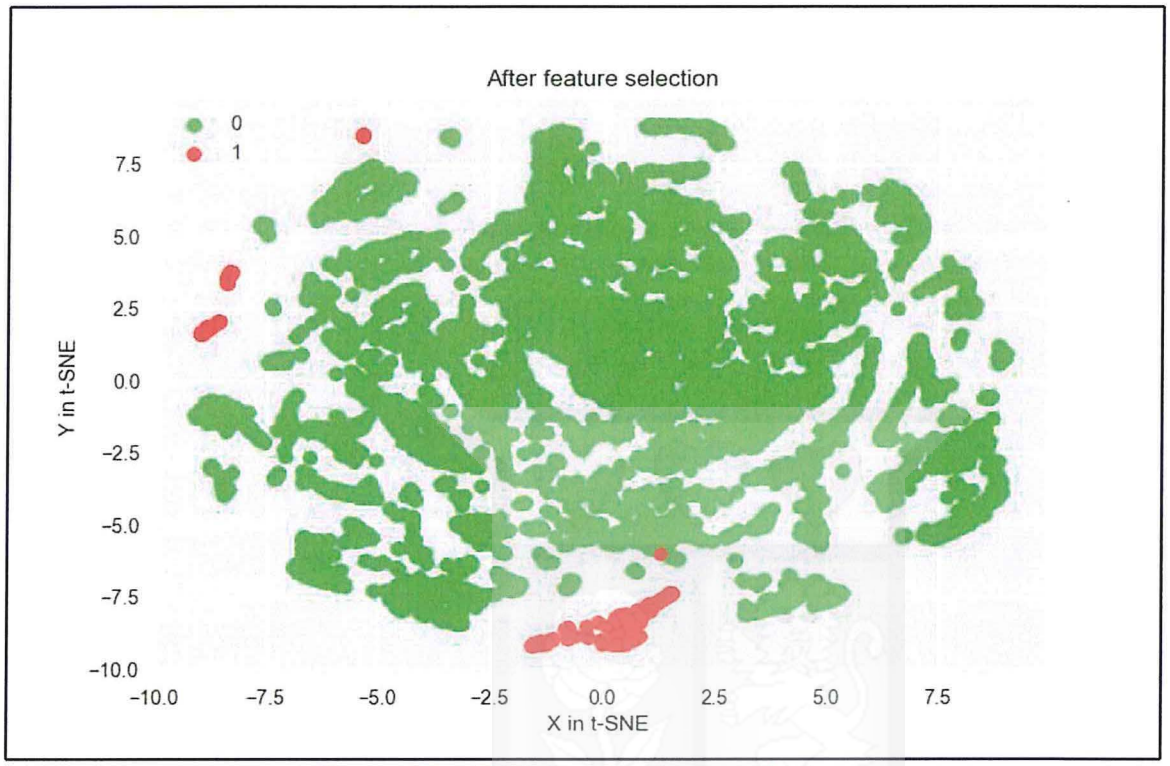

Figure 12: t-SNE plot after feature selection

After the feature reduction we see better groupings of fraudulent transactions forming with fewer scattered fraudulent transactions. This leaves us to conclude that the feature selection was useful.

4.3. Data partitioning

From the entire dataset, 3 different subsets are extracted: training, validation and testing. Rather than split the data linearly each subset is set to contain a proportion of fraudulent transactions since it forms the underrepresented class. The subsets were then divided to form the training, testing and validation set accounting for 60%, 20% and 20% of the entire sample respectively.

Subset	No. of obs.	% Fraud
Training	170,884	0.1726
Test	56,961	0.1668
Validation	56,962	0.1790

Table 3: Data partitioning

4.4. Handling Class Imbalance

Imbalance due to rare instances is representative of domains where minority class examples are very limited, i.e., where the target concept is rare (He & Garcia, 2009). In such a scenario, the lack of representative data makes learning difficult regardless of the between-class imbalance (Weiss, 2004).

To overcome sampling methods are used in imbalanced learning applications consisting of the modification of an imbalanced data set by some mechanisms in order to provide a balanced distribution. The mechanics of random oversampling follow naturally from its description is utilized by adding a set E sampled from the minority class: for a set of randomly selected minority examples in S_{min}, augment the original set S by replicating the selected examples and adding them to S. In this way, the number of total examples in S_{min} is increased by $|E|$ and the class distribution balance of S is adjusted accordingly. This provides a mechanism for varying the degree of class distribution balance to any desired level. The oversampling method is simple to both understand and visualize (He & Garcia, 2009).

According to He & Garcia (2009) studies have shown that for several base classifiers, a balanced data set provides improved overall classification performance compared to an imbalanced data set. These results justify the use of sampling methods for imbalanced learning. However, they do not imply that classifiers cannot learn from imbalanced data sets; on the contrary, studies have also shown that classifiers induced from certain imbalanced data sets are comparable to classifiers induced from the same data set balanced by sampling techniques. This phenomenon has been directly linked to the problem of rare cases as is the case with this study. It was found that the oversampling technique did not aid in improved classifier accuracy and in fact led to overfitting. Such is the consequence brought about by oversampling since it simply appends replicated data to the original data set, multiple instances of certain examples become tied, leading to overfitting(Zheng, 2012). In particular, overfitting in oversampling occurs when classifiers produce multiple clauses in a rule for multiple copies of the same example which causes the rule to become too specific; although the training accuracy will be high in this scenario, the classification performance on the unseen testing data is generally far worse (Holter, Ackner, & Porter, 1989).

CHAPTER 5: RESULTS & CONCLUSION

This section gives the details of the overall results of the proposed models and compares the performance of the Feed Forward Neural Network (FFNN) and Support Vector Machine (SVM) model.

5.1. Feed Forward Neural Network

The FFNN machine learning algorithm used in this experiment was implemented in python. Neural Networks tend to be complex and more difficult to implement because they require one to make considerations around the network specifications to use and the values of the network parameters for training.

As opposed to jumping straight into the multilayer neural network it was thought prudent to first check the results of the single layer FFNN. Thereafter, additional layers would be incorporates in a series of experiments until the optimal number of hidden layers was found. The figures below illustrate the FFNN architecture selected in these experiments.

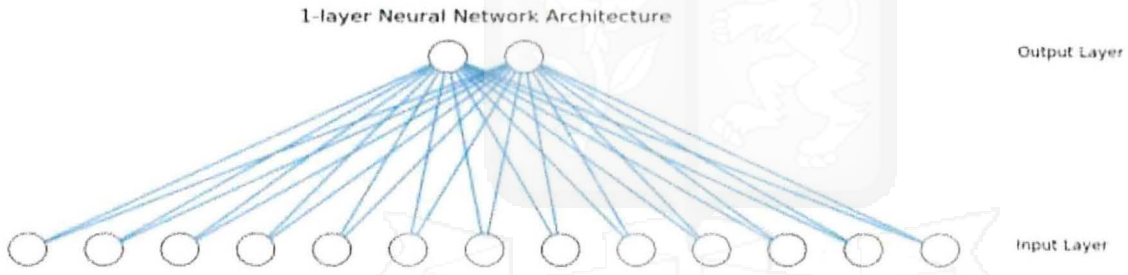

Figure 13: 1-layer Neural Network

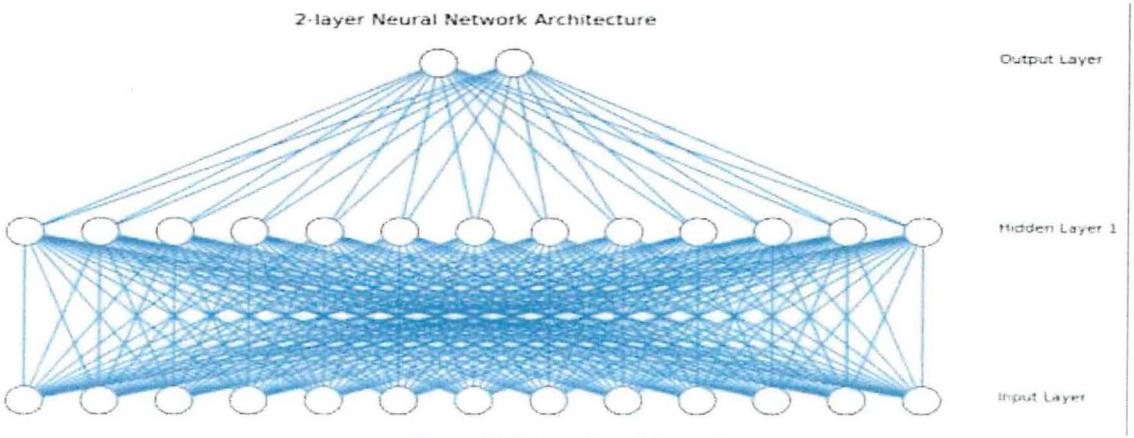

Figure 14: 2-layer Neural Network

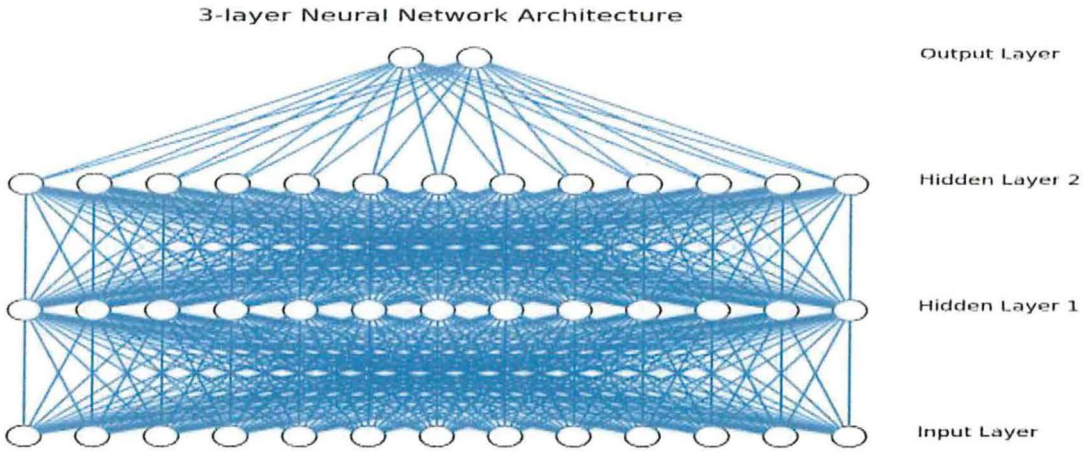

Figure 15: 3-layer Neural Network

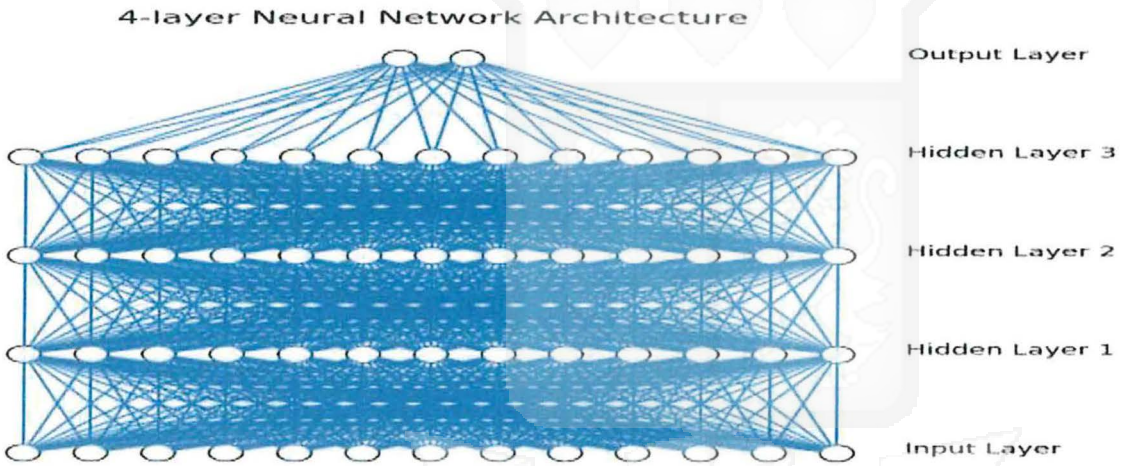

Figure 16: 4-layer Neural Network

Fully connected networks were used containing 13 input neurons and 2 output neuron. During initial experiments the network architecture that contained as many hidden nodes as input nodes achieved the best results; this same architecture was therefore used in the final FFNN experiments.

Using tensorflow to build the predictive model the initial parameters were set to:

```
## Parameters
max training_epochs = 1000
batch_size = 2048
learning_rate = 0.005
```

The training epochs represent the number of full training cycles on the training set used to update the weights. For batch training all of the training samples pass through the learning algorithm simultaneously in one epoch before weights are updated. The batch size represents the number of training examples in one forward or backward pass. Appropriate selection of batch size is

important because the higher the batch size, the more memory space required. Similarly when examining the learning rate, too low a rate makes the network learn very slowly. Too high a learning rate on the other hand makes the weights and objective function diverge, so there is no learning at all. In contrast to a fixed learning rate, gradually decreasing the learning rate, after each epoch would assist in decreasing training time (LeCun, Bottou, Orr, & Müller, 2012). During training the `Adam` optimization was used and the resulting weights stored. It is based on adaptive estimates of lower-order moments and in recent years has gained popularity due to its low memory requirements, invariance to diagonal rescaling of the gradients as well as its suitability for problems containing large data with multiple parameters (Kingma & Ba, 2014). The results of the experiment are summarized in the table below:

	1-layer	*2-layer*	*3-layer*	*4-layer*
Training Accuracy	97.58%	98.31%	99.72%	99.26%
		Testing		
Testing Accuracy	97.67%	98.34%	99.68%	99.22%
Testing F1-Score	98.82%	99.16%	99.84%	99.61%
Testing ROC_AUC	94.53%	94.39%	90.48%	90.49%
Precision	7.43%	8.33%	32.37%	15.17%
Sensitivity	91.38%	90.43%	81.25%	81.72%
Specificity	97.68%	98.35%	99.71%	99.25%
False Negative Rate	8.62%	9.57%	18.75%	18.28%
False Discovery Rate	92.57%	91.67%	67.63%	84.83%
G-measure	43.39%	43.62%	54.89%	44.67%
		Validation		
Validation Accuracy	97.52%	98.24%	99.68%	99.19%
Validation F1-Score	98.74%	99.11%	99.84%	99.59%
Validation ROC_AUC	94.44%	91.36%	93.41%	94.80%
Precision	5.00%	8.09%	34.11%	17.28%
Sensitivity	91.36%	84.47%	87.13%	90.38%
Specificity	97.53%	98.26%	99.70%	99.21%
False Negative Rate	8.64%	15.53%	12.87%	9.62%
False Discovery Rate	95.00%	91.91%	65.89%	82.72%
G-measure	41.75%	43.62%	58.42%	49.43%
Time	≈ 2 min	≈ 2 min	≈ 8 min	≈ 7 min

Table 4: Summary of FFNN results

Additional information on the training accuracy and cost is summarized diagrammatically in the appendix as well as the confusion matrix of the different models.

Generally, it is seen from the results that the 1-layer and 2-layer FFNN exhibit similar characteristics and performed better in identifying Fraud as indicated by the high sensitivity and

low false negative rate. On the contrary the 3-layer and 4 layer FFNN showed better performance in identifying Normal transactions and produced less false alarms as is seen by the higher specificity and lower false discovery rate. Wiese & Omlin (2009), highlight that fraud detection systems suffer from unacceptable false alarm rates, making the probability of annoying legitimate customers much higher than that of actually detecting fraud which adds to the cost of detecting fraud. With this in mind it is clear that the 1-layer, 2-layer and 4-layer FFNNs would be less suitable in fraud detection as they all give unacceptably high false positives relative to the 3-layer FFNN.

In finding the best model among the four one would thus consider whether the model can correctly identify fraud and minimize false alarm rates. Considering these criteria the best classifier was found to be the 3-layers FFNN. This is further backed by the results of F1-score and G-measure.

5.1.1. Deep Neural Networks

Given the poor performance recorded by the 'shallow' FFNNs an attempt was made to take into account current trends in *Deep Learning*. Deep neural networks have achieved remarkable results in computer vision, natural language processing, and speech recognition areas (Yuan, Wu, Li, & Lu, 2017). Deep Neural Networks (DNN) are larger neural networks organized into layers of neurons, corresponding to successive representations of the input data. The difference is in the NN architecture which has a higher number of hidden layers.

Deep Neural Network			
Testing		***Validation***	
Testing Accuracy	99.95%	***Validation Accuracy***	99.94%
Testing F1-Score	99.97%	***Validation F1-Score***	99.97%
Testing ROC_AUC	93.46%	***Validation ROC_AUC***	90.46%
Precision	80.81%	***Precision***	85.86%
Sensitivity	86.96%	***Sensitivity***	80.95%
Specificity	99.97%	***Specificity***	99.98%
False Negative Rate	13.04%	***False Negative Rate***	19.05%
False Discovery Rate	19.19%	***False Discovery Rate***	14.14%
G-measure	83.86%	***G-measure***	83.39%
	Time	≈ 1 min	

Table 5: Summary of Deep Neural Network Results

The results clearly show that DNN yields much better classification results than the more 'shallow' NN. It yielded a precision and sensitivity score above 80% which means it was able to better identify fraudulent transactions while giving low false alarms. The high specificity score indicates its the ability to identify normal transactions. In addition the G-measure score was 84% giving a much better model from a practical stand point.

5.2. Support Vector Machine

The performance of an SVM depends on the kernel used as well as the kernel parameters and cost set by the user. Since SVM training guarantees to always find the maximum margin achievable for a given set of parameters, it is fruitless to conduct multiple trial runs with the same parameters and time should rather be spent finding the most appropriate kernel and its parameters. Unfortunately, no real satisfying heuristic exists with which to compute a kernel's parameters and in most cases the best one can do is to guess (Wiese & Omlin, 2009). The search for the best parameters, however, can be done in a structured manner. Through a combination of cross-validation and grid-search, a popular method, one can obtain a good estimate of the cost parameter (C) and the kernel parameter (γ) when dealing with radial basis function (RBF) kernels. Cross-validation can also help to prevent over fitting of the training set (Hsu et al., 2003).

In a v-fold cross-validation, the training set is divided into v parts and the classifier sequentially trained on v-1 of the subsets, and then tested on the remaining subset. This way, each subset is tested once and each training instance is predicted once, and the cross-validation accuracy is therefore the percentage of training examples correctly classified (Wiese & Omlin, 2009).

For grid-search in order to estimate the best parameters for a given classification problem and kernel, a sequence of different parameter values are utilized during training and those giving the best cross-validation accuracy are picked. In case of the RBF kernel, the parameter values consist of (C, γ) pairs.

In this experiment the best parameters for the model using grid search and across a 10-fold cross validation was (1.25,'auto') and the process took approximately 15 min.

SVM			
Testing		**Validation**	
Precision	100.00%	***Precision***	100.00%
Sensitivity	95.96%	***Sensitivity***	97.96%
Specificity	100.00%	***Specificity***	100.00%
False Negative Rate	4.04%	***False Negative Rate***	2.04%
False Discovery Rate	0.00%	***False Discovery Rate***	0.00%
Testing F1-Score	97.94%	***Testing F1-Score***	98.97%
Test ROC_AUC	97.98%	***Validation ROC_AUC***	98.98%
G-Measure	97.98%	***G-Measure***	98.98%
	Time	≈ 39 sec	

Table 6: Summary of SVM results

The results obtained show that the classifier was able to correctly identify 100% of Normal transactions hence no false alarms. The model was also able to identify more than 95% of the Fraud occurrences in the different test runs.

5.3. Conclusion

It is clear from the comparison between the 3-layer NN and the DNN that the latter is a much better classifier. It is able to give almost twice as good fraud detection capability while minimizing the costs associated with misclassification (Value of False Positives of the 3-layer FFNN ≈ €30,000 while that of DNN was ≈ €3,000).

Overall looking at the results obtained from both the Feed Forward Neural Networks and Support Vector Machine it is clear that SVM outperforms the rest. The advantages the SVM model presents is identifying the correct class of transactions with the lowest false discovery rate. In addition it is easier to set up with fewer parameters required during training.

However, the DNN presents the much sorted appeal of reduced computational time and memory in training and testing the model. This makes it harder to pick a clear winner.

CHAPTER 6: DISCUSION

This paper examined the performance of two advanced Machine Learning techniques, Feed Forward Neural Networks and Support Vector Machines. SVM demonstrated overall better performance across the various measures although the Deep Learning technique employed gives somewhat impressive results with much less computational time.

An important factor contributing to the better performance of the SVM was the careful selection of features and optimization of parameters used. Particularly, it was noted that although exploratory data analysis and feature selection was a time consuming step, if such effort was not applied then giving the relative performance of the model would be misleading.

When considering practical implementation it is observed that standard SVM training has $O(n^3)$ time and $O(n^2)$ space complexities, where n is the training set size. It is thus computationally infeasible on very large data sets. However, existing literature attempts to addresses this problem with examples such as the Core Vector Machine (CVM) algorithm which can be used with nonlinear kernels and has a time complexity that is *linear* in n and a space complexity that is *independent* of n. Tsang, Kwok, & Cheung (2005), through experiments on large toy and real world data sets demonstrated that the CVM is as accurate as existing SVM implementations, but is much faster and can handle much larger data sets. This presents a useful subject for further investigation.

The FFNN despite having better computational efficiency in terms of speed and memory produced a much higher false positive rate which would make it unsuitable from a practical usage standpoint. The Deep Neural Network employed came closest to the performance accuracy of the SVM. In theory the FFNN has an $O(n^2)$ time complexity during training with $f(n)$ given by the number of epochs, observations, features and neurons.

In addition, although this paper examines credit card fraud detection the scope of application of this techniques can be extended to other binary classification problems in different sectors. For example: determining if a patient has a disease or not, information retrieval, mobile money fraud detection (which was the initial topic that motivated this paper) etc. As a final remark rather than considering these models in isolation an attempt could be made to combine the two through ensemble learning and thus harness the benefits therein derived.

REFERENCES

Akande, K. O., Owolabi, T. O., Twaha, S., & Olatunji, S. O. (2014). Performance comparison of SVM and ANN in predicting compressive strength of concrete. *IOSR Journal of Computer Engineering*, *16*(5), 88–94.

Baesens, B., Van Vlasselaer, V., & Verbeke, W. (2015). *Fraud analytics using descriptive, predictive, and social network techniques: a guide to data science for fraud detection*. John Wiley & Sons.

Bargen, D. (2013, March 26). Programming a Perceptron in Python - blog.dbrgn.ch. Retrieved September 19, 2017, from https://blog.dbrgn.ch/2013/3/26/perceptrons-in-python/

Bhattacharyya, S., Jha, S., Tharakunnel, K., & Westland, J. C. (2011). Data mining for credit card fraud: A comparative study. *Decision Support Systems*, *50*(3), 602–613.

Bivens, A., Palagiri, C., Smith, R., Szymanski, B., Embrechts, M., & others. (2002). Network-based intrusion detection using neural networks. *Intelligent Engineering Systems through Artificial Neural Networks*, *12*(1), 579–584.

Bolton, R. J., & Hand, D. J. (2002). Statistical fraud detection: A review. *Statistical Science*, 235–249.

Bolton, R. J., Hand, D. J., & others. (2001). Unsupervised profiling methods for fraud detection. *Credit Scoring and Credit Control VII*, 235–255.

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory* (pp. 144–152). ACM. Retrieved from http://dl.acm.org/citation.cfm?id=130401

Buczak, A. L., & Guven, E. (2016). A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials*, *18*(2), 1153–1176.

Burges, C. J. (1998). A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, *2*(2), 121–167.

Cannady, J. (1998). Artificial neural networks for misuse detection. In *National information systems security conference* (pp. 368–81). Retrieved from http://pld.cs.luc.edu/courses/intrusion/fall05/cannady.artificial_neural_networks_for_misuse_detection.pdf

Chang, C., & Lin, C. (2012). LIBSVM: a library for support vector machines," 2001. Software available at http://www. csie. ntu. edu. tw/~ cjlin/libsvm. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.552.5513

Commission, F. T., & others. (2013). Mobile privacy disclosures: Building trust through transparency. *USA: Federal Trade Commission*.

Corbo, J., Giovine, C., & Wigley, C. (n.d.). Applying analytics in financial institutions' fight against fraud | McKinsey & Company. Retrieved July 31, 2017, from http://www.mckinsey.com/business-functions/mckinsey-analytics/our-insights/applying-analytics-in-financial-institutions-fight-against-fraud

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297.

Cristianini, N., & Shawe-Taylor, J. (2000). *An introduction to support vector machines*. Cambridge University Press Cambridge. Retrieved from http://library02.embl.de/InmagicGenie/DocumentFolder/TableOfContents_H898.pdf

Davis, J., & Goadrich, M. (2006). The relationship between Precision-Recall and ROC curves. In *Proceedings of the 23rd international conference on Machine learning* (pp. 233–240). ACM.

Haykin, S. (2009). *Neural networks and learning machines* (3. ed). New York: Pearson.

He, H., & Garcia, E. A. (2009). Learning from Imbalanced. Retrieved September 23, 2017, from https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=Learning+from+Imbalanced+Data+Haibo+He&btnG=

Hilas, C. S., & Mastorocostas, P. A. (2008). An application of supervised and unsupervised learning approaches to telecommunications fraud detection. *Knowledge-Based Systems*, *21*(7), 721–726.

Holter, R., Ackner, L., & Porter, B. (1989). Concept Learning and the Problem of Small Disjuncts. Retrieved September 23, 2017, from https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=R.C.+Holte%2C+L.+Acker%2C+and+B.W.+Porter%2C+%E2%80%9CConcept+Learning+and+the+Problem+of+Small+Disjuncts%2C%E2%80%9D+Proc.+Int%E2%80%99l+J.+Conf.+Artificial+Intelligence%2C+pp.+813-818%2C+1989.&btnG=

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*(5), 359–366.

Horvitz, E., & Mulligan, D. (2015). Data, privacy, and the greater good. *Science*, *349*(6245), 253–255.

Hsu, C.-W., Chang, C.-C., Lin, C.-J., & others. (2003). A practical guide to support vector classification. Retrieved from http://www.datascienceassn.org/sites/default/files/Practical%20Guide%20to%20Support%20Vector%20Classification.pdf

Hu, W., Liao, Y., & Vemuri, V. R. (2003). Robust Support Vector Machines for Anomaly Detection in Computer Security. In *ICMLA* (pp. 168–174). Retrieved from http://web.cs.ucdavis.edu/~vemuri/papers/rvsm.pdf

Husain, R., & Vohra, R. (2017). APPLYING MACHINE LEARNING IN THE FINANCIAL SECTOR. *International Education and Research Journal*, *3*(1). Retrieved from http://ierj.in/journal/index.php/ierj/article/view/637

IFC, & MasterCard Foundation. (2016). *DIGITAL FINANCIAL SERVICES AND RISK MANAGEMENT*.

Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, *349*(6245), 255–260.

Keerthi, S. S., & Lin, C.-J. (2003). Asymptotic behaviors of support vector machines with Gaussian kernel. *Neural Computation*, *15*(7), 1667–1689.

Kingma, D., & Ba, J. (2014). Adam: A method for stochastic optimization. *ArXiv Preprint ArXiv:1412.6980*. Retrieved from https://arxiv.org/abs/1412.6980

Kroon, S., & Omlin, C. W. (2004). Getting to grips with support vector machines: Theory. *South African Statistical Journal*, *282*, 159–172.

LeCun, Y. A., Bottou, L., Orr, G. B., & Müller, K.-R. (2012). Efficient backprop. In *Neural networks: Tricks of the trade* (pp. 9–48). Springer. Retrieved from http://link.springer.com/chapter/10.1007/978-3-642-35289-8_3

Lee, Y., Lin, Y., & Wahba, G. (2001). Multicategory support vector machines. In *Proceedings of the 33rd Symposium on the Interface*. Retrieved from https://pdfs.semanticscholar.org/3a6d/9e4c0ef0ac8f3b1f4cc2da4463ef6fc473cd.pdf

Lin, C.-T., & others. (1996). *Neural fuzzy systems: a neuro-fuzzy synergism to intelligent systems*. Prentice hall PTR. Retrieved from https://ir.nctu.edu.tw/handle/11536/107399

Lippmann, R. P., Fried, D. J., Graf, I., Haines, J. W., Kendall, K. R., McClung, D., … others. (2000). Evaluating intrusion detection systems: The 1998 DARPA off-line intrusion detection evaluation. In *DARPA Information Survivability Conference and Exposition, 2000. DISCEX'00. Proceedings* (Vol. 2, pp. 12–26). IEEE. Retrieved from http://ieeexplore.ieee.org/abstract/document/821506/

Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., & Byers, A. H. (n.d.). Big data: The next frontier for innovation, competition, and productivity | McKinsey & Company. Retrieved July 31, 2017, from http://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/big-data-the-next-frontier-for-innovation

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, *5*(4), 115–133.

Meyer, D., Leisch, F., & Hornik, K. (2003). The support vector machine under test. *Neurocomputing*, *55*(1), 169–186.

Minotaur™ Fraud Detection Software. (n.d.). Retrieved May 30, 2017, from https://swipx.com/apps/minotaur-fraud-detection-software/

Mitchell, T. M. (1997). Machine learning. 1997. *Burr Ridge, IL: McGraw Hill*, *45*(37), 870–877.

Ngai, E. W. T., Hu, Y., Wong, Y. H., Chen, Y., & Sun, X. (2011). The application of data mining techniques in financial fraud detection: A classification framework and an academic review of literature. *Decision Support Systems*, *50*(3), 559–569.

Patidar, R., Sharma, L., & others. (2011). Credit card fraud detection using neural network. *International Journal of Soft Computing and Engineering (IJSCE)*, *1*(32–38). Retrieved from http://ijsce.org/attachments/File/NCAI2011/IJSCE_NCAI2011_025.pdf

Purcell, C. (2016, May 1). Neural Technologies and Safaricom win Global Telecoms Innovation Award. Retrieved May 27, 2017, from https://www.neuralt.com/33/278/neural-technologies-and-safaricom-win-global-telecoms-innovation-award

Ripley, B. D. (2002). *Statistical data mining*. Springer-Verlag, New York. Retrieved from http://www.stats.ox.ac.uk/pub/bdr/SDM2002/DM2002.pdf

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*(6), 386.

Schölkopf, B., & Smola, A. J. (2002). *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press. Retrieved from

https://books.google.com/books?hl=en&lr=&id=y8ORL3DWt4sC&oi=fnd&pg=PR13&dq=B.+Sch%C2%A8olkopf,+A.+Smola,+Learning+with+Kernels,+MIT+Press,+2002.&ots=bKBW8AW2Dy&sig=GFqQRAgK9fHI0ydwCwWY2BnLTR4

Sharma, R. K., Kalita, H. K., & Borah, P. (2016). Analysis of machine learning techniques based intrusion detection systems. In *Proceedings of 3rd International Conference on Advanced Computing, Networking and Informatics* (pp. 485–493). Springer. Retrieved from http://link.springer.com/chapter/10.1007/978-81-322-2529-4_51

Shon, T., & Moon, J. (2007). A hybrid machine learning approach to network anomaly detection. *Information Sciences*, *177*(18), 3799–3821.

Sudjianto, A., Nair, S., Yuan, M., Zhang, A., Kern, D., & Cela-Díaz, F. (2010). Statistical methods for fighting financial crimes. *Technometrics*, *52*(1), 5–19.

Tsang, I. W., Kwok, J. T., & Cheung, P.-M. (2005). Core vector machines: Fast SVM training on very large data sets. *Journal of Machine Learning Research*, *6*(Apr), 363–392.

Van Vlasselaer, V., Eliassi-Rad, T., Akoglu, L., Snoeck, M., & Baesens, B. (2016). Gotcha! Network-based fraud detection for social security fraud. *Management Science*. Retrieved from http://pubsonline.informs.org/doi/abs/10.1287/mnsc.2016.2489

Waltz, D. L. (1988). The prospects for building truly intelligent machines. *Daedalus*, 191–212.

Weiss, G. M. (2004). Mining with rarity: a unifying framework. *ACM SIGKDD Explorations Newsletter*, *6*(1), 7. https://doi.org/10.1145/1007730.1007734

Wiese, B., & Omlin, C. (2009). Credit card transactions, fraud detection, and machine learning: Modelling time with LSTM recurrent neural networks. In *Innovations in neural information paradigms and applications* (pp. 231–268). Springer. Retrieved from http://link.springer.com/chapter/10.1007/978-3-642-04003-0_10

Yuan, S., Wu, X., Li, J., & Lu, A. (2017). Spectrum-based deep neural networks for fraud detection. *ArXiv Preprint ArXiv:1706.00891*.

Zheng, S. (2012). QBoost: Predicting quantiles with boosting for regression and binary classification. *Expert Systems with Applications*, *39*(2), 1687–1697.

APPENDIX

Testing

1-layer

		Predicted	
		Fraud	Normal
Actual	Fraud	106	10
	Normal	1,320	55,525

2-layer

		Predicted	
		Fraud	Normal
Actual	Fraud	85	9
	Normal	936	55,931

3-layer

		Predicted	
		Fraud	Normal
Actual	Fraud	78	18
	Normal	163	56,702

4-layer

		Predicted	
		Fraud	Normal
Actual	Fraud	76	17
	Normal	425	56,443

Validation

1-layer

		Predicted	
		Fraud	Normal
Actual	Fraud	74	7
	Normal	1,407	55474

2-layer

		Predicted	
		Fraud	Normal
Actual	Fraud	87	16
	Normal	988	55,871

3-layer

		Predicted	
		Fraud	Normal
Actual	Fraud	88	13
	Normal	170	56,691

4-layer

		Predicted	
		Fraud	Normal
Actual	Fraud	94	10
	Normal	450	56,408

Testing

DNN

		Predicted	
		Fraud	Normal
Actual	Fraud	80	12
	Normal	19	56,850

Validation

DNN

		Predicted	
		Fraud	Normal
Actual	Fraud	85	20
	Normal	14	56,843

Testing

SVM		Predicted	
		Fraud	**Normal**
Actual	**Fraud**	95	4
	Normal	0	56,863

Validation

SVM		Predicted	
		Fraud	**Normal**
Actual	**Fraud**	96	2
	Normal	0	56,863

Table 7: Summary of Results in Confusion Matrix

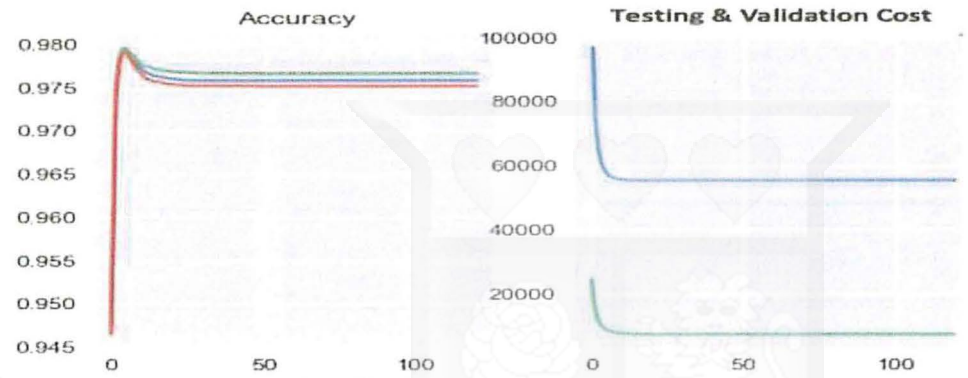

Figure 17: 1-layer training accuracy and cost

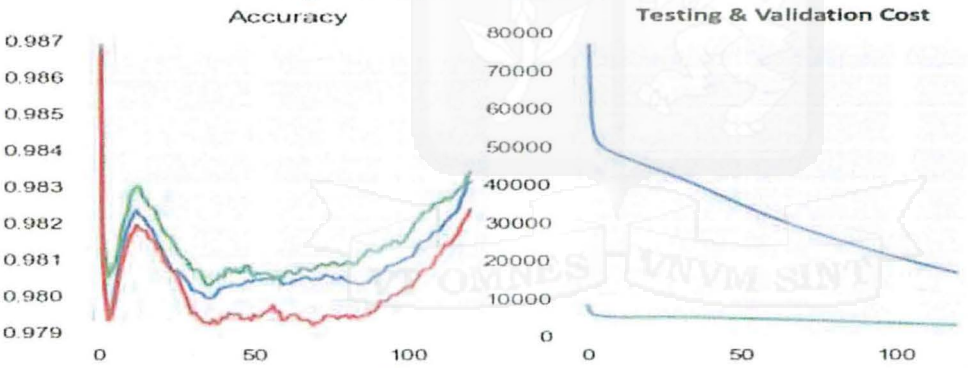

Figure 18: 2-layer training accuracy and cost

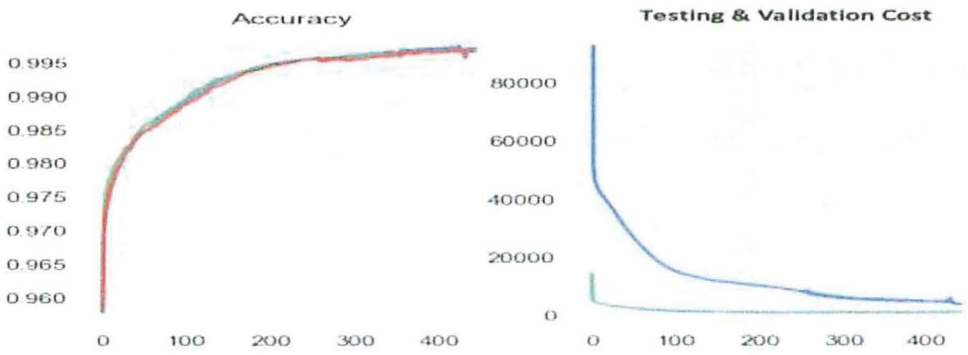

Figure 19: 3-layer training accuracy and cost

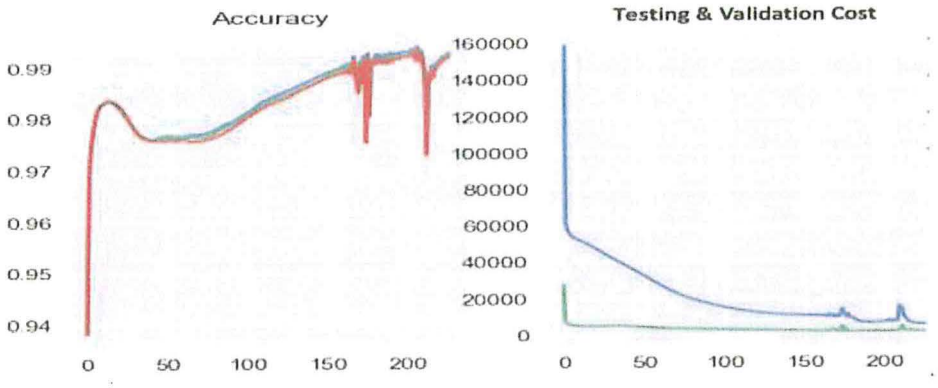

Figure 20: 4-layer training accuracy and cost

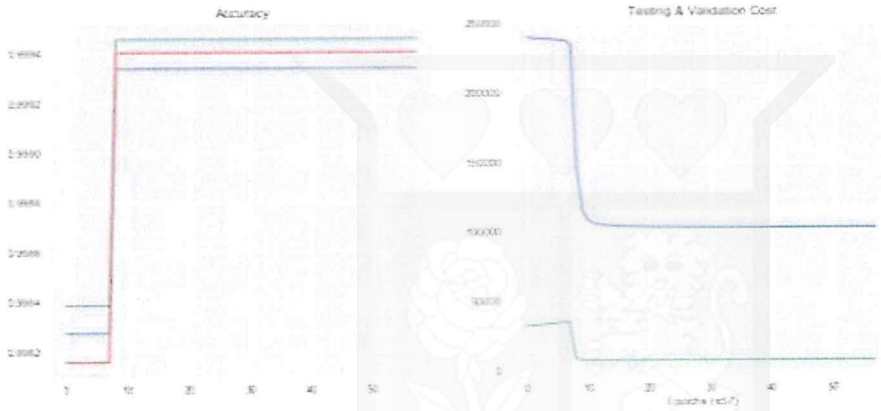

Figure 21: Deep Neural Network accuracy and cost