Improving Fraud Detection in Credit Card Transactions Using Autoencoders and Deep Neural Networks

by Sanjay Nakharu Prasad Kumar

B.E. in Electronics and Telecommunications, June 2008, University of Pune
M.B.A. in Finance, April 2013, Symbiosis International University
M.S. in Information Systems, May 2017, The University of Texas at Arlington

A Praxis submitted to

The Faculty of
The School of Engineering and Applied Science
of The George Washington University
in partial fulfillment of the requirements
for the degree of Doctor of Engineering

August 31, 2022

Praxis directed by

Donald V. Widener
Professorial Lecturer of Engineering Management and Systems Engineering

Amir Etemadi
Assistant Professor of Engineering and Applied Science

The School of Engineering and Applied Science of The George Washington University certifies that Sanjay Nakharu Prasad Kumar has passed the Final Examination for the degree of Doctor of Engineering as of August 8, 2022. This is the final and approved form of the Praxis.

Improving Fraud Detection in Credit Card Transactions Using Autoencoders and Deep Neural Networks

Sanjay Nakharu Prasad Kumar

Praxis Research Committee:

Donald V. Widener, Professorial Lecturer of Engineering Management and Systems Engineering, Praxis Co-Director

Amir Etemadi, Assistant Professor of Engineering and Applied Science, Praxis Co-Director

John Kamp, Professorial Lecturer of Engineering Management and Systems Engineering, Committee Member

Dedication

My dissertation is dedicated to the mentors at my institution who have guided me throughout the process of writing this dissertation. In addition to sharing academic knowledge, they have provided valuable advice throughout the whole process.

Acknowledgments

I would like to express my sincere gratitude to my supervisors, Dr. Don Widener and Dr. Amir Etemadi, for their continuous support throughout my doctoral study, for their patience, motivation, and immense knowledge. Their guidance helped me throughout the time of the research and writing of this thesis. I could not imagine having better supervisors and mentors for my D.Eng. study.

Abstract of Praxis

Improving Fraud Detection in Credit Card Transactions Using Autoencoders and Deep Neural Networks

In recent years, the use of credit cards has increased significantly due to digitization and the emergence of cashless transactions. There has been a huge jump in fraud in credit card transactions across banks, credit unions, and other financial institutions across the globe. Financial institutions and credit card companies are now required to detect credit card fraud in real time to prevent further losses.

This research proposed a method to improve fraud detection using unsupervised deep learning classifier autoencoders and deep neural networks. It is difficult to train machine learning models for credit card fraud detection because of the class imbalance. To build an effective classifier or predictive model, it is necessary to balance the number of legitimate and fraudulent transactions. Data resampling was applied to the dataset through random undersampling and random oversampling, specifically the synthetic minority oversampling technique (SMOTE) and adaptive synthetic (ADASYN) technique. Since resampling introduced noise in the data, an unsupervised deep learning–based autoencoder algorithm was used for denoising. Using the denoised dataset, a deep neural network was built to classify transactions in the sampled dataset as normal or fraudulent.

To implement the autoencoder, this research used Google's TensorFlow library, which is part of the TensorFlow framework. Model performance was evaluated using metrics such as precision, recall, F1 score, and area under the curve–receiver operating characteristic. The study model outperformed existing models in the industry.

Table of Contents

List of Figures

Page

List of Tables

List of Acronyms

ADASYN	Adaptive synthetic
ANN	Artificial neural networks
AUC	Area under the curve
DNN	Deep neural network
FN	False negative
FP	False positive
PCA	Principal component analysis
ROC	Receiver operating characteristic
SMOTE	Synthetic minority oversampling technique
SVM	Support vector machine
TN	True negative
TP	True positive

CHAPTER 1:

INTRODUCTION

1.1. Background

Society is becoming more and more reliant on the internet, making credit cards a convenient and essential convenience. Transactions involving credit cards have become the de facto standard for internet e-commerce. As a result of the popularity of credit cards and their widespread use, there have been many reports of credit fraud on a global scale. Identity theft and data breaches are the two most common threats. To prevent fraud and unauthorized access to information, a more secure system is needed. A more secure system is urgently needed since credit cards generally seem vulnerable to various risks.

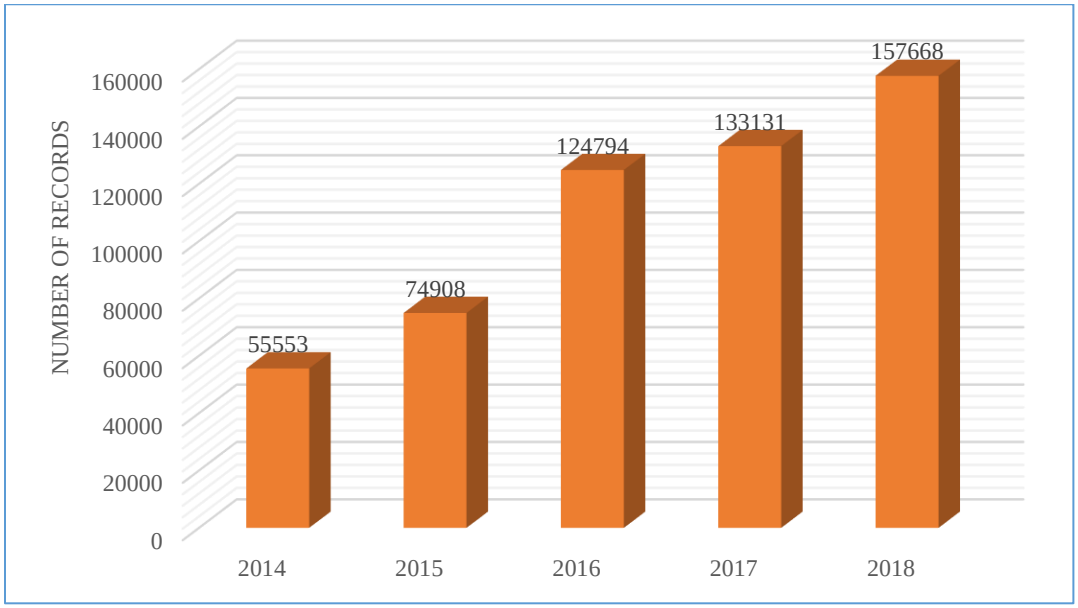
Worldwide, countless forms of cybercrime occur every second. In July 2019, the most extensive cyberattack in the history of the credit card industry occurred with the compromise of the Capital One database (Silver, 2019). Approximately $28.58 billion was lost through card fraud in 2020 and $32.20 billion in 2021; it is estimated that $49.32 billion will be lost in 10 years (Nilson Report, 2019). In 2020, financial card fraud in the United States accounted for 22.19% of the total volume of fraud worldwide, making up 33.57% of the gross losses. In 2020, fraud involving financial cards reached $19.13 billion in all other countries, which comes to 5.89¢ per $100 in total volume (Nilson Report, 2019).

Increasing amounts of financial loss are incurred through credit/debit cards in the world every year, contributing to a potential financial crisis. Therefore, advanced credit card fraud detection systems are needed to address this urgent situation.

1.1.1. Credit Card Fraud: Statistics

Over the past decades, several systems have attempted to improve credit card security, but nevertheless the amount of losses and the number of cybercrime reports have increased dramatically (Figure 1.1). The credit card system's security was significantly improved to make it more reliable and secure after cyberattacks negatively affected many individuals and companies. According to a consumer feedback survey in 2018 and an updated survey in 2019, 85.71% of victims felt worried, angry, and frustrated, while 83.7% felt violated and 69.4% felt unsafe (Shift Credit Card Processing, 2021). Credit card fraud is one of most significant problem the world is facing today.

Figure 1.1
Credit Card Fraud Reported in the United States

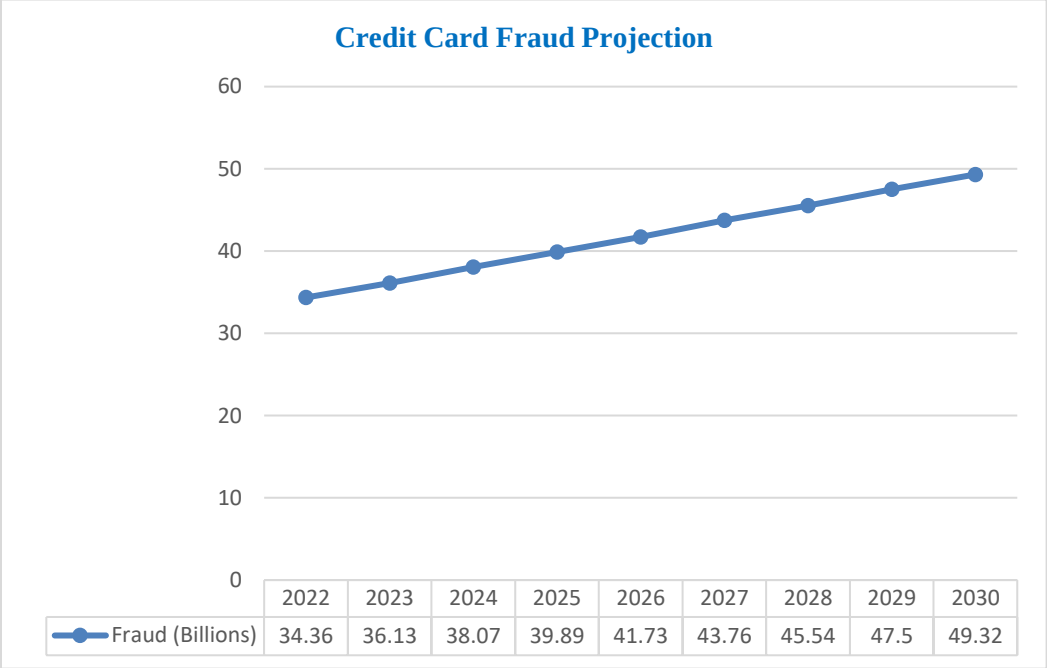


Note: Data from Shift Credit Card Processing (2021) and Krishna (2021).

According to the Nilson Report (2019), by 2025, total payment card volume worldwide is projected to be $60.583 trillion, with gross card fraud globally expected to

be $39.89 billion. While total credit card volume is expected to increase to $79.140 trillion by 2030, the fraud amounts are projected to increase at a lower rate; nevertheless, the total amount of fraud in billions is projected to continue to rise, even as the cents per $100 volume of transactions is expected to decline from 6.75¢ to 6.23¢ between 2022 and 2030 (Figure 1.2). Losses to fraud in the United States are projected to reach $12.51 billion in 2025.

Figure 1.2
Credit Card Fraud Projection

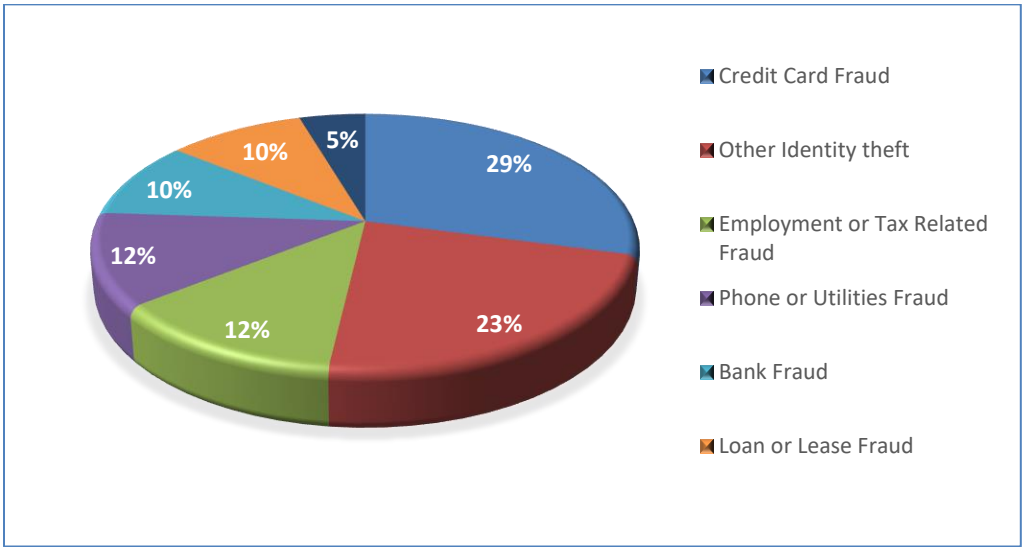


Note: Based on data from the Nilson Report (2019).

Statistics show that 2020 may prove to be one of the biggest fraud years in history. During the pandemic, there was a drastic increase in the number of identity theft cases, as well as a surge in government benefits fraud. The number of fraud reports increased by 120,000 over the past years. Credit card fraud increased at a much slower

rate than other forms of identity theft. As shown in Figure 1.3, identity theft accounted for 29% of all fraud reports in 2018.

Figure 1.3
Identity Theft Fraud Reports



Note: Based on data from Shift Credit Card Processing (2021).

In the last year, the number of credit card fraud cases has doubled, and that number is expected to continue to rise. The stolen funds have an impact on both consumers and businesses since they can significantly damage business operations and financial systems. In a world where unauthorized financial activities are rampant, businesses may have difficulty receiving regular payments and may even go out of business.

1.1.2. Types of Credit Card Fraud

Several methods can be used to classify credit card fraud, including application fraud and behavioral fraud. Application fraud involves applying for a credit card using false personal identification, while behavioral fraud involves obtaining the cardholder's

credentials to use a pre-existing credit card. There are five basic types of financial fraud: theft or loss of cards, fraud committed online or through the internet, fraud committed by merchants, the use of bogus cards, and the theft of cards through exportation.

According to industry experts, several methods can be used to classify fraudulent transactions:

1. **Loss/theft of cards (1% of fraudulent transactions):** It is common for fraudsters to target elderly cardholders to steal the card by stealing their PIN number. This is a case where credit card information is not sold in the open market or through the dark web; rather, the one who obtains the card or PIN is the one who commits the fraud.
2. **Credit cards not received (<1% of fraudulent transactions):** It has been reported that credit cards have been stolen during production or during postal delivery. One solution is the bank can ask a customer to collect a credit card from a bank branch and activate it there. Another solution is to require direct contact with the bank in the activation process for a new card.
3. **ID theft/stolen identity (least used method):** An individual can obtain a credit card and its details using a false or stolen identification card.
4. **Bogus credit cards (<10% of fraudulent transactions):** Fraud of this type is planned on an international scale and takes place by hacking databases, obtaining original credit cards, and then replicating them on fake plastic cards. As result, fraudsters are copying and reproducing the magnetic stripe data on the cards. This type of fraud used to predominate in the past but has been partially addressed

through chip technology. Due to limitations of contactless cards, they are less appealing to fraudsters.

5. **Fraud committed with cards not present (>90% of fraudulent transactions**): In the majority of cases, credit card fraud occurs on websites like Amazon, eBay, and other online shops that accept credit card payments. Often, international crime operations use the dark web or black market to sell credentials such as card numbers, expiration dates, and card verification codes. It has been reported that Ticketmaster and United Airlines have both been affected by data breaches in the past year. A major obstacle to preventing this type of fraud is the fact that companies do not report breaches of their data due to public relations issues.

1.2. Research Motivation

This research paper aimed to improve the accuracy of existing algorithms that predict fraud in financial institutions using machine learning algorithms so that if a transaction is not authorized by the account owner, it will not go through. There is a very limited number of academic studies on the underlying causes of fraud, including costs, system loopholes, and how and why it occurs. Even fewer studies address how fraud can be detected and avoided, including identifying the causes. There is a large demand for antifraud experts. If organizations want to avoid fraud and protect their reputations, they must devote considerable resources to preventing fraud. It may be hard for smaller organizations to set up antifraud units due to insufficient resources. Companies need to hire private investigation firms to deal with fraud, which can be quite expensive. Banks and financial institutions can benefit from the use of machine learning and deep learning.

1.3. Problem Statement

By 2025, fraudulent credit card charges will increase to $35.31 billion because fraudsters are coming up with more innovative and foolproof methods to beat current fraud detection systems (Nilson Report, 2019; Silver, 2019). Fraud losses increase each year both in the United States and worldwide. Card-based payment systems worldwide suffered gross fraud losses of 6.78¢ for every $100 of total volume in 2019. Fraud losses in the US were 10.25¢ per $100 based on data collected from the Federal Reserve.

1.4. Thesis Statement

The thesis statement for this study was as follows: *Using supervised and unsupervised deep learning classifiers to build predictive models that use historical transactions and suspicious activity can significantly increase fraud detection accuracy*.

Fraudsters' behavior is constantly changing, so there are no constant patterns of fraud. Fraudsters learn about new technology through which they can execute online fraud. The purpose of this study is to describe a hybrid technique that combines supervised and unsupervised methods with the aim of improving the accuracy of fraud detection systems.

1.5. Research Objectives

The purpose of the study was to examine the effectiveness of an advanced prediction model that utilizes artificial neural networks and autoencoders in order to detect fraud in credit card transactions. Detailed objectives are as follows:

1. Apply synthetic minority oversampling technique (SMOTE) method to improve minority class detection accuracy.

2. To access ensemble techniques-based model to improve classification Accuracy
3. To develop an advanced predictive model to detect fraud in credit card transactions using a hybrid approach of autoencoders and deep neural networks.

1.6. Research Questions and Hypotheses

This study addressed three research questions, based on the objectives of the study:

RQ1: When datasets are highly imbalanced, what is the best sampling approach and how can it be applied?

RQ2: Can ensemble technique–based classifiers like random forest give better results than classical classification models like logistic regression?

RQ3: Can a hybrid method that combines supervised and unsupervised techniques improve fraud detection accuracy?

There were three corresponding hypotheses:

H1: The SMOTE technique for class imbalance provides statistically better results than Undersampling and ADASYN.

H2: Ensemble technique–based classifiers like random forest give better AUC score than classification models like logistic regression and decision tree.

H3: A hybrid method that combines supervised and unsupervised techniques improves fraud detection accuracy.

1.7. Scope of Research

Using statistical methods and machine learning techniques, this study aimed to identify a variety of methods that can be applied to detect credit card fraud in different

situations. This study proposed a two-stage model for identifying credit card fraud in the context of credit card transactions. To make a trustworthy fraud detection system, it is critical to understand and learn complex associations among the attributes that make up the transaction. To address this issue, the proposed model featured an autoencoder, which was used in the first stage of processing. By doing this, it is possible to eliminate unwanted noise from sampled datasets that may have been introduced during the sampling process. The dataset obtained was then fed as input to a deep neural network at the second stage for classification. The experiment was performed on a benchmarked dataset to evaluate the results.

1.8. Research Limitations

The detection of credit card fraud is extremely challenging but is also a general problem that needs to be addressed. Due to limited data visibility, current fraud solutions produce significant false-positives due to insufficient data in entrusted transactions.

This study used only 2 days of credit card transactions, creating a rather small dataset. Because of its small size, it contained a relatively low proportion of fraudulent transactions—approximately 0.0017%.

Since principal component analysis was applied to the Kaggle dataset before it was released to the public, it is difficult to understand why fraudulent transactions were classified as fraud. As it is unclear what constitutes fraud, it cannot be determined whether fraudulent transactions were affected by geography, for instance. Since this dataset pertained only to European cardholders, it is assumed that the model is limited to European banks.

1.9. Organization of Praxis

The praxis consists of five chapters. This first chapter has provided background information that informs the study's motivation and has outlined the research objectives, questions, and hypotheses and the study's scope and limitations. Chapter 2 reviews the relevant literature concerning credit card fraud and the banking system, including a variety of research papers on credit card fraud detection. It also discusses machine learning algorithms and use of deep learning and neural networks for detecting credit card fraud. Chapter 3 details the methodology for the study, and Chapter 4 presents the results and analyses. The praxis closes with Chapter 5, which summarizes the results and provides some insight into future work.

CHAPTER 2:

LITERATURE REVIEW

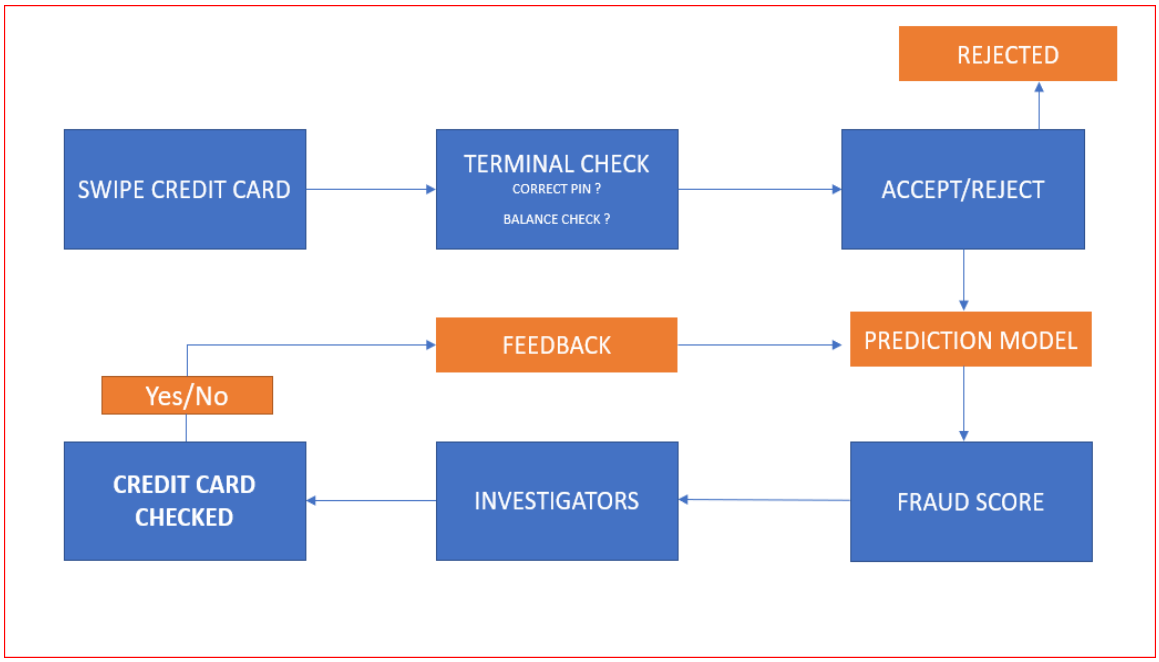
2.1. Introduction

Many financial institutions have invested significant time and effort into developing fraud detection programs to combat the threat of credit card fraud. They have assembled teams of people from around the world. Many companies and researchers are actively working to mitigate the problems that can arise when developing a fraud detection system, such as class imbalance issues, overlapping classes, changes in fraud behaviors, and many other hidden factors that may appear during this process. This chapter describes how credit card fraud is detected, reviews research efforts to improve the process, and discusses various machine learning and neural network algorithms being used in fraud detection.

2.2. Credit Card Fraud Detection Process

A credit card fraud detection process involves verifying credit card payments through a terminal check, and then scoring them using a predictive model to detect the most suspicious transactions if they are not rejected. As the investigators evaluate the alerts (fraud or genuine transactions), they provide feedback that can be used to enhance the predictive model (Figure 2.1).

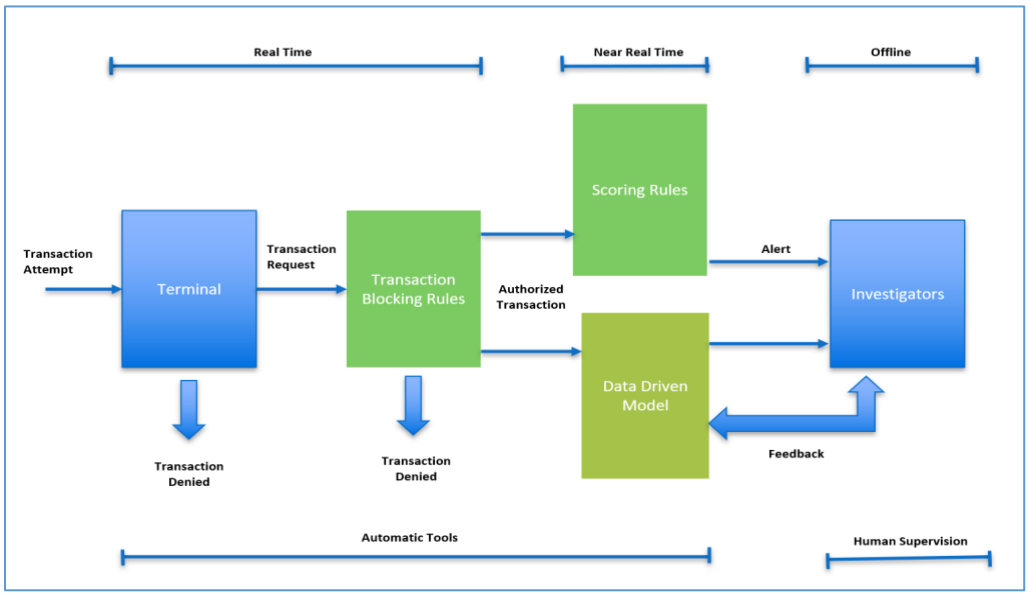
Figure 2.1
Credit Card Fraud Detection Flow



Note: Image source: Suryanarayana et al. (2018).

In practice, credit card fraud detection involves two steps: blocking fraudulent transactions and checking fraudulent cards (Figure 2.2). There is a difference between the two processes in terms of time requirements; time is a very important component here. The blocking time is very limited since the cardholder/customer is waiting to determine whether the transaction is authorized or approved or not, while the process of checking can take a considerable amount of time.

Figure 2.2
Credit Card Fraud Detection System



Note: Image Source : Pozzolo, A.D., & Bontempi, G. (2015

As soon as a credit card payment/transaction has been authorized, the fraud detection system is activated. Machine learning algorithms are used together with rules developed by industry experts to create this detection system. Whenever the system triggers an alert, the alert is examined by experts. A credit card company may contact the cardholders in some cases so that they can verify the transaction in order for the fraud to be traced (Pozzolo, 2015).

In reality, it is not possible to hold or block every transaction that has a suspicion of fraud, and it is not possible to verify each and every transaction due to the cost and human resources needed to carry out this activity. To solve this problem, an automated process is needed for the fraud detection system. This addresses the issue and saves money on operations. A fraud prevention system like this can prevent financial institutions from losing a lot of money (Lucas, 2019).

2.3. Studies Analyzing Credit Card Fraud Detection

Many studies and investigations have been conducted about credit card fraud detection over the past few years. This section reviews some past research in this field.

Some studies have focused on process. For example, Jiang et al. (2018) proposed a system that involved collecting the transactions made by cardholders, applying the behavior of cardholders to the aggregated transactions, classifying the consolidated dataset, training the model, and finally testing the model. When abnormal behavior arises, the system is notified by feedback mechanisms, which allow it to be alerted on time.

Studies have also addressed online authentication procedures. Gupta and Johari (2011) reviewed and analyzed the risks associated with online transactions due to the sharing of confidential credit card information on the web. They provided a framework that can be used for both online and onsite credit card transactions. The framework has some advantages, such as enhancing privacy and providing a level of security that is not possible with other frameworks.

Makki and colleagues (2019) described the impact that credit card fraud has on banks and other financial institutions. In an effort to eliminate this loss, numerous academic studies have been conducted. Abundant methods are available, but many of them are expensive, time consuming, and labor intensive. Most of them also have poor efficiency rates. According to Makki et al. (2019), unclassified datasets are largely to blame for inaccurate results. The imbalanced dataset ultimately leads to an inaccurate predictive model and financial losses. Based on metrics like area receiver operating characteristics (ROC) and accuracy, logistic regression, decision tree algorithms, support vector machine (SVM), neural networks, and artificial neural networks (ANN) were the

best algorithms. To ensure that these models were as accurate as possible, they were trained using the balanced dataset (Makki et al., 2019).

According to Sadgali et al. (2019), it is becoming more and more common for people to prefer digital and paperless transactions, so online and mobile transactions will continue to grow in popularity. The number of transactions has reached the millions; each is subject to a type of fraud. Machine learning models for fraud detection have been designed, developed, and analyzed extensively. To identify fraud in card transactions, a comparison has been made between different machine-learning algorithms to choose the most suitable classifier for detecting fraud (Sadgali et al., 2019).

Maes et al. (2002) used Bayesian classifiers along with a deep neural network (DNN) to detect fraud in credit card transactions. Despite the smaller training time, Bayesian networks are very efficient in detecting fraud; however, ANN has a quicker learning process and helps to detect fraud more quickly.

Kumar et al. (2019) developed a machine learning classifier or algorithm for detecting fraud in credit card transactions using random forest (ensemble method). Basically, a random forest algorithm uses a decision tree to detect fraud in credit card transactions and a confusion matrix for calculating performance. Accuracy of 90% was achieved by the proposed system (Kumar et al., 2019).

Prusti and Rath (2019) developed models using decision tree, k-nearest algorithm, extended learning machine, multiple layer perceptron, and SVM to prevent fraud in transactions. To facilitate efficient data exchange across heterogeneous platforms, they used two web-based protocols: simple object access protocol and representational state transfer. Based on performance metrics, they compared the results of five models. “SVM

performed better than other algorithms by 81 percent, but their hybrid system had a better accuracy by 82 percent" (Prusti & Rath, 2019).

Similarly, Trivedi and colleagues (2020) used a feedback mechanism to detect fraud. As a result of feedback, the classifier was able to detect more errors. The study compared different methods using an imbalanced dataset. The models included random forest, naive Bayes, decision trees, ANN, logistic regression, SVM, and gradient boosting classifier strategies. Random forest outperformed other classifiers and achieved an accuracy of 94.99%. Despite this, random forests take a very long time to run to produce accurate results (Trivedi et al., 2020).

For fraud detection, Sohony et al. (2018) "proposed an ensemble learning method based on the ratio of fraud to normal transactions." The researchers discovered that random forest is one of the best algorithms for detecting fraud instances, and neural networks provide a higher level of accuracy. Also, they used a large dataset with real transactions to detect fraud on card transactions. They indicated that random forest combined with neural networks provide better accuracy (Sohony et al., 2018).

In the credit card domain, Lucas et al. (2020) evaluated methods of feature engineering in detecting fraud using hidden Markov models. Based on past transactions, hidden Markov models estimate the probability that a current transaction is fraud or not. A random forest is used to detect fraud based on the results. Furthermore, the model provides improved results in ROC curves for the evaluation of credit card fraud by considering the issue of missing values (Lucas et al., 2020).

The next sections review these techniques in more detail, adding notes about additional studies using each method. Section 2.4 reviews machine learning; Section 2.5, deep learning; and Section 2.6, autoencoders.

2.4. Machine Learning Techniques

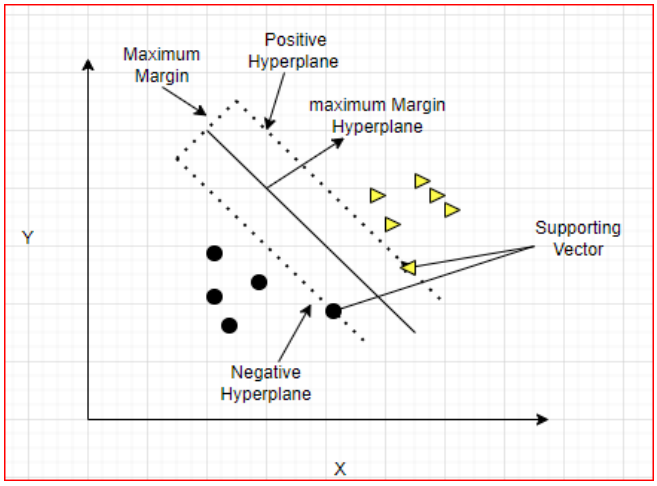
Fraudulent transactions can now be detected more easily and reliably using machine learning. To learn patterns behind fraudulent behavior, a machine learning model is trained using a historical dataset. Real-time fraud detection can then be achieved by applying a model to filter out fraudulent transactions. Depending on the method of learning, machine learning can be classified as three types: supervised learning, unsupervised learning, and reinforcement learning. This section describes five different machine learning algorithms used in classification.

2.4.1. Support Vector Machine

SVMs fall under the category of supervised learning, which means that to properly train them, the algorithm needs a label. SVM can be used for classification as well for regression (Adepoju et al., 2019).

An SVM is an algorithm that can be used to find a hyperplane that is capable of helping categorize data in a way that is highly useful. Hence, the main objective is to find the optimal plan that can effectively differentiate two classes. In Figure 2.3, the data point on the right side represents a genuine transaction, while the data point on the left side represents a fraudulent transaction. A maximizing margin distance will allow future data points to be classified more confidently in the future, leading to more accurate classification.

Figure 2.3
Support Vector Machine



Note: Image source: Analytics Vidhya.

Sahin and Duman (2011b) evaluated decision trees and SVMs for detecting fraud in credit card transactions. They found that decision trees are able to detect fraud more accurately than SVMs. After training the SVM machine with a large amount of data, the SVM gives the same accuracy as well as ROC score (Sahin & Duman, 2011b).

Based on the findings of Bhattacharyya and his team (2011), fraud in credit card transactions can be detected by three algorithms: random forest, logistic regression, and SVMs. They used different sampling techniques to resample the data and then used the same sampled data to train these classifiers. They demonstrated that logistic regression is more accurate and more reliable than other algorithms when it comes to accuracy and recall (Bhattacharyya et al., 2011).

2.4.2. Naive Bayes

The naive Bayes algorithm was presented by John and his team in 1995. The model is based on probability and allows predictions to be made simultaneously for

several classes at the same time. The Bayes theorem is the foundation for this algorithm. In the process of deciding, conditional probability is one of the approaches used. This algorithm is the result of combining several algorithms that adhere to the same principle; there is no single algorithm in this model. The algorithm can be trained with little cost, which is one of its advantages (Awoyemi et al., 2017).

The naive Bayes theorem makes a decision based on highest probabilities. The naive Bayes classifier ignores the fact that features depend on one another. A naive Bayes algorithm is a supervised algorithm that can be expressed as shown in Equation 2.1.

$$P(c|x) = \frac{P(x|c)P(c)}{P(x)} \qquad 2.1$$

$$P(c|X) = P(x_1\,|c) \times P(x_2\,|c) \ldots P(x_n\,|c) \times P(c)$$

The Bayes' theorem states that the posterior likelihood of an outcome *c* under certain special conditions *x* is determined by the posterior likelihood P (x|c). To calculate the later probability of a result, we need to compare the earlier probability with its later probability, using the Bayes theorem, by comparing P (c|x) and P (c), which is the probability ratio between the two. It is an assumption that each factor will contribute to a particular outcome independently; hence, the naive Bayes theorem must be regarded as naive.

In their research, Phua et al. used back-propagation, the naive theorem, and the decision tree to resample skewed data partitions because of minority oversampling with replacement. There have been some studies showing that the use of partitioned numerical data, naive classifiers, and back-propagation classifiers can be used to reduce costs (Maes et al., 2002).

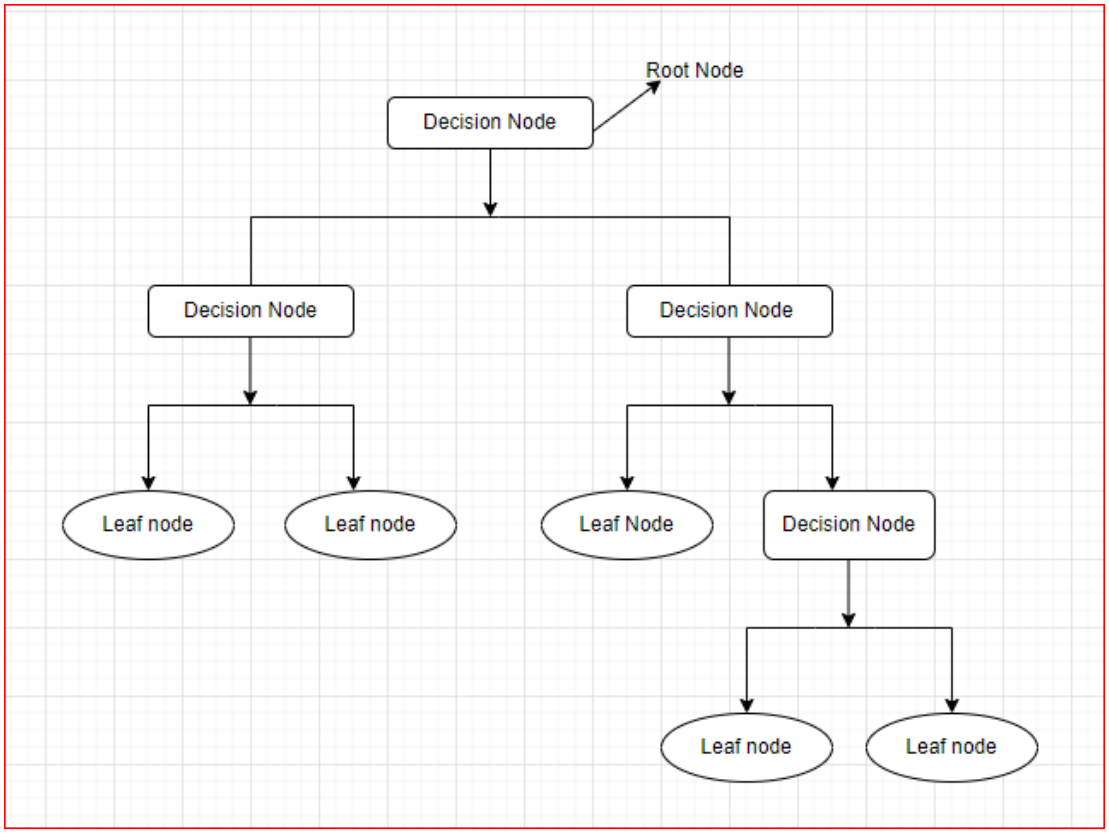
Sherly (2012) discussed how supervised data mining techniques can be used to prevent fraud. Decision trees, neural networks, and naive Bayes classifiers were evaluated. According to the study, neural network–based classifiers are suitable for large databases since they take a long time to train. Compared to other algorithms, Bayesian classifiers can be trained more quickly but learning is slower.

2.4.3. Decision Trees

In machine learning, decision trees are used as a means of structuring algorithms as part of the machine learning process. With the help of a decision tree algorithm, dataset features can be split based on a cost function. The "pruning method helps to remove branches from a decision tree that are using irrelevant features before it is optimized for finding the best solution" (Quinlan, 1986).

Among the types of machine learning problems, classification problems are most frequently solved by decision trees. The method involves supervised machine learning, where the model is trained to determine the type of object the data represents based on the type of data being presented. The decision to make strategic splits heavily impacts the accuracy of a tree. Decision trees use multiple algorithms like Gini to split nodes into subnodes to increase homogeneity. A node's purity increases in relation to a target variable. The goal is to select the split that results in the most homogeneous subnodes based on all available variables (Figure 2.4).

Figure 2.4
Decision Tree

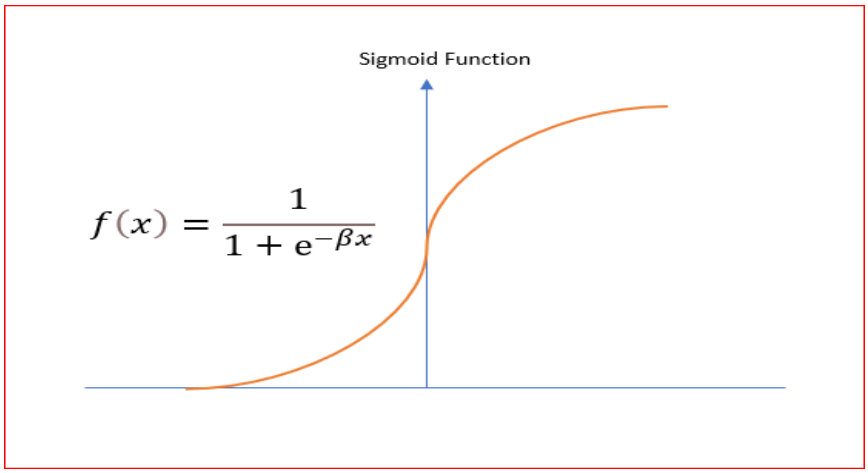


Save et al. (2017) indicated that they "used decision trees in conjunction with Luhn's algorithm and Hunt's algorithm to detect fraudulent transactions." In this case, the authorized user's billing address matched his shipping address. For a legitimate transaction to proceed, these addresses should match. Otherwise, the transaction is classified as suspicious. The study referred to this as "outlier detection" and concluded that this method is reliable and has low false alarm rates.

2.4.4. Logistic Regression

There are independent variables that are used in calculating the probability a certain event will occur, such as the possibility of fraud, for example. It would be expected that the dependent variable would range from 0 to 1 due to the probability of the outcome (Sahin & Duman, 2011a). After determining a linear relationship from a dataset, logistic regression introduces nonlinearity through the sigmoid function (Figure 2.5).

Figure 2.5
Sigmoid Function



Logistic regression uses a logit transformation that is applied to the odds as a measure of success or failure—namely, the probability of success divided by the probability of failure. Also called the logistic function, the natural logarithm of odds can be expressed as follows.

$$logit(pi) = \frac{1}{1+e^{-Pi}} \qquad 2.2$$

Ng and Jordan (2002) compared the efficiency of logistic regression with that of naive Bayes. Logistic regression was observed to have lower error rates than naive

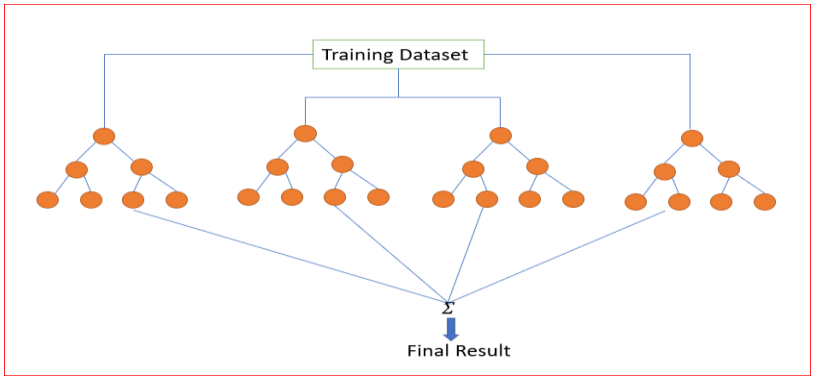
Bayes. This helps to ensure that logistic regression converges. Furthermore, logistic regression has been shown to perform better on smaller datasets than naive Bayes (Ng & Jordan, 2002).

According to Sahin and Duman (2011a), ANN and logistic regression are two techniques that can help to detect credit card fraud with highly skewed data. They found that ANN were more effective than logistic regression in the classification problems. Because of insufficient sampling of the work in logistic regression classifiers, as training data increase, they can fit the data as a result of working with a larger set of data (Sahin & Duman, 2011a).

2.4.5. Random Forest

The random forest model is an aggregate classifier that combines many decision trees. The idea behind using multiple trees is to train them sufficiently so that they each participate in the structure of a model (Figure 2.6). The results will be combined after the tree is constructed. Random forest can be used to solve both classification and regression problems (Xuan et al., 2018).

Figure 2.6
Random Forest



Note: Image source: IBM Cloud Education (2020a).

As part of the fraud detection process, Lakshmi and Kavilla (2018) examined algorithms such as decision trees, random forests, and logistic regressions using the Kaggle dataset. As an alternative to Python, the R programming language was used as part of the research process. To evaluate the model, the researchers used different parameters. In their study, they found that logistic regression had an accuracy of 90.0%, random forest had an accuracy of 95.5%, and decision tree had an accuracy of 94.3% (Lakshmi & Kavilla, 2018). Thus, random forest had the best accuracy.

2.4.6. Comparison of Machine Learning Techniques for Fraud Detection

Table 2.1 assesses machine learning credit card fraud detection techniques in terms of performance, cost, and time.

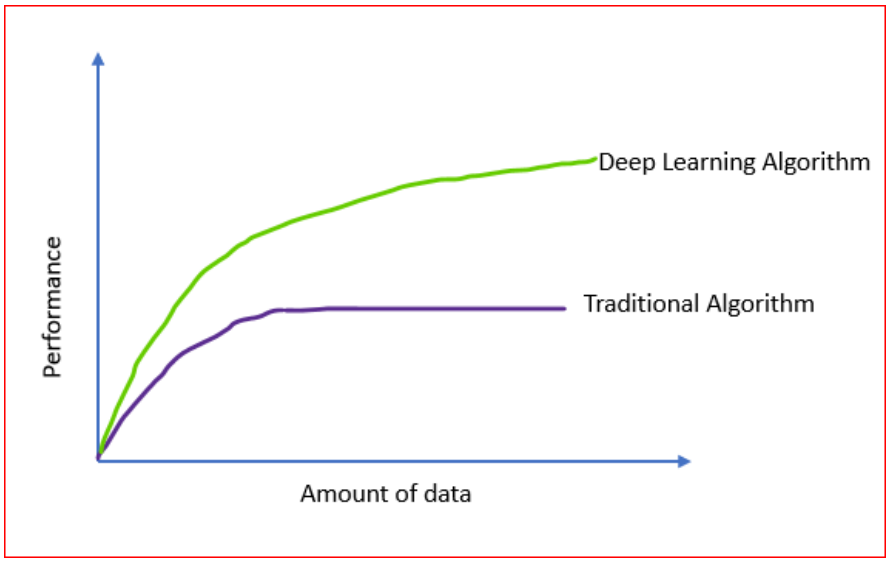
Table 2.1
Comparison of Machine Learning Techniques for Credit Card Fraud Detection

Fraud detection technique	Advantages	Disadvantages
K-nearest neighbor algorithm	Useful for determining anomalies in target instances	Has memory limitations
Random forest	Work well with regression and classification	Takes a long time for training
Neural network	By learning previous behavior, can detect credit card fraud in real time	Has lower performance when it picks up things that aren't suitable for detecting credit card fraud
Decision tree	Can handle nonlinear credit card transaction as well	Have many types of input features and cannot detect fraud in real time during transactions
Deep learning	Cn analyze and learn large amounts of unsupervised data	Not all algorithms are covered in the deep learning library
Logistic regression	Is more accurate with larger sample sizes and has a low cost	Depends on more variables for better results

2.5. Deep Learning Techniques

A machine learning system and a deep learning system are two of the most important components of artificial intelligence. Machine learning algorithms can be used for simple tasks, and data structures should be simple. Machine learning includes deep learning as one of its subdomains. Deep learning is capable of performing more complex tasks than other machine learning techniques. In Figure 2.7, when the learning process stops even after providing more data, deep learning keeps learning over time.

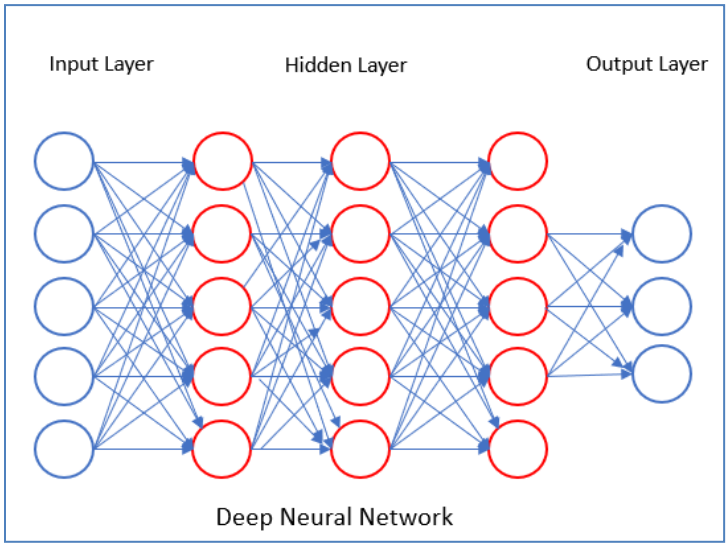
Figure 2.7
Deep Learning Compared with Traditional Machine Learning



When a deep learning classifier consists of more than three layers, it is called deep learning. The hidden layers of a complex deep learning algorithm can contain more than 100 layers. The basic building blocks of deep learning are neurons, which are the smallest units of the brain. Using neural networks inspired by biology, computers are able to learn by using information found in observational data. One of the most significant

features of deep learning is the hidden layer. With the increase in the number of hidden layers, it is possible to achieve better learning results (Figure 2.8).

Figure 2.8
Deep Neural Network



The other advantage of deep learning is its ability to select the features automatically, whereas in the case of a conventional machine learning process, the features must be selected manually before the model can be trained.

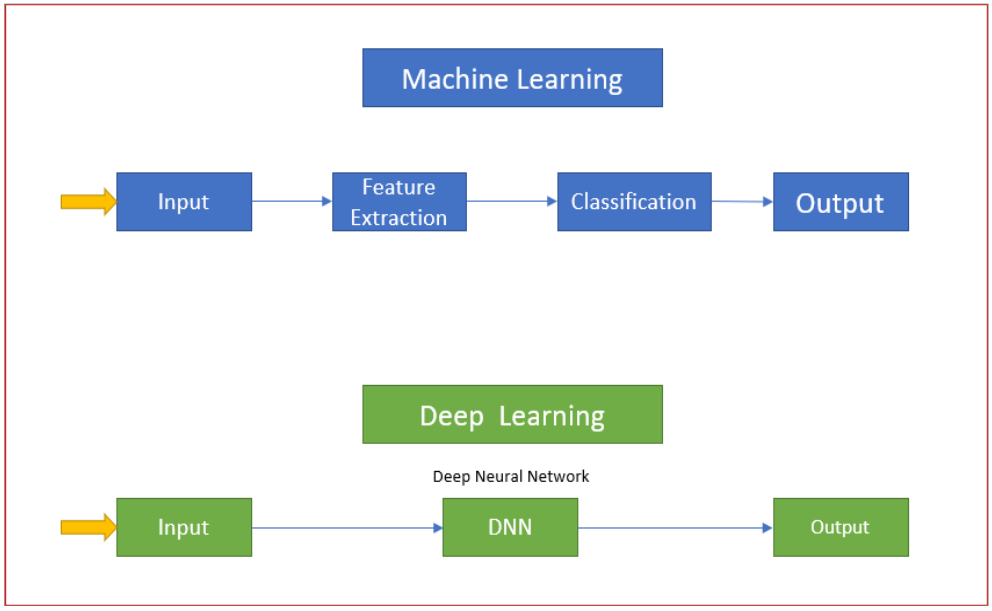
2.5.1. Difference Between Machine Learning and Deep Learning

IBM defined deep learning as "a subfield of machine learning that has at least 3 layers (input, hidden, and output) and it forms a neural network system" (IBM Cloud Education, 2020b). This network will resemble or behave like the human brain with these three layers. As the number of layers increases, the network will attain a higher level of accuracy. It is possible to use deep learning algorithms to analyze unstructured datasets, such as images and text. During the learning process, the following steps are involved:

- Analyze the data and feed it into the algorithm. By performing feature extraction in this step, additional information is provided to the model.
- Train the model using this data.
- Evaluate and implement it.

The model can be used to perform an automated predictive task. Moreover, deep learning algorithms are designed to scale with data, whereas shallow learning algorithms tend to converge over time. After adding more examples and training data to a neural network, shallow learning approaches plateau. In the case of deep learning algorithms, efficiency can be greatly improved as a result of the large volume of data and the many hidden layers embedded within them. Figure 2.9 illustrates the differences between machine learning and deep learning.

Figure 2.9
Machine Learning vs Deep Learning

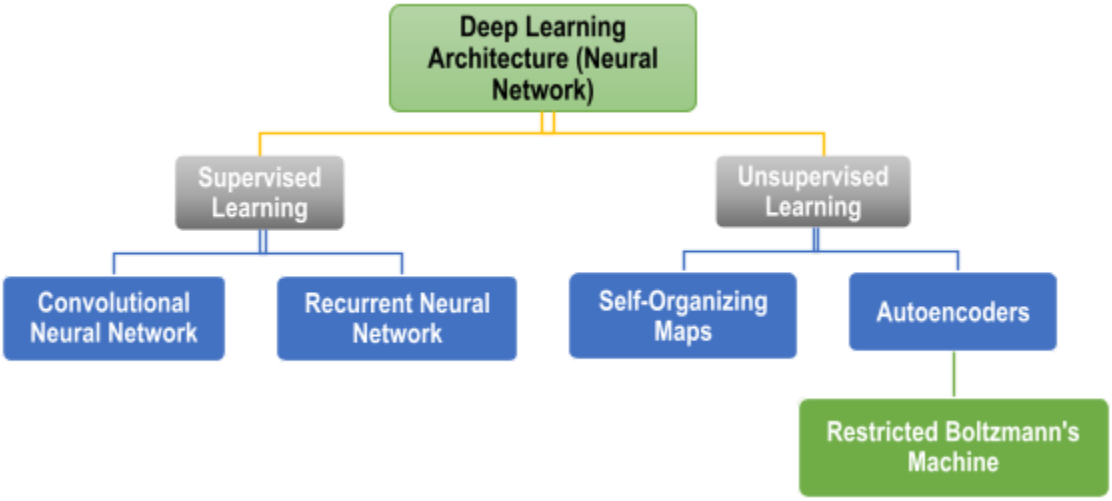


Note: Image source: ProjectPro (2022).

2.5.2. Deep Learning Methods

In the field of deep learning, a wide range of architectures and algorithms are used, as shown in Figure 2.10. Among the approaches, developed over 20 years of research, long short-term memory and convolutional neural networks are two of the earliest and most popular. As this research focused on autoencoders, those are explored in detail in the following section.

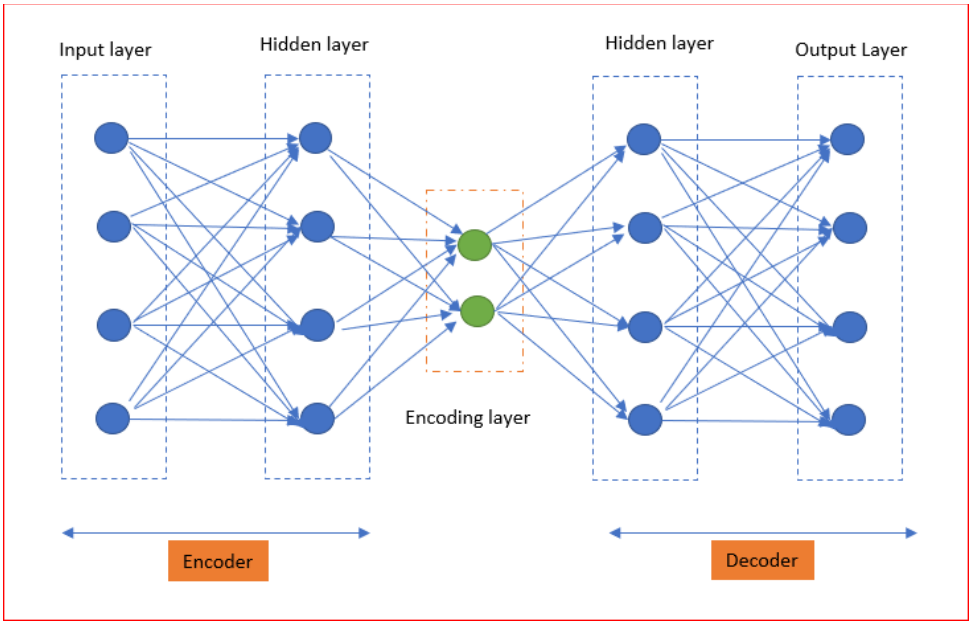
Figure 2.10
Methods of Deep Learning



2.6. Autoencoders

The autoencoder is an unsupervised deep learning method. It is a simple feed-forward network where the number of inputs equals the number of outputs. As a result of passing the input to the low dimension code, the low dimension code can reconstruct the input. It is also known as a latent space representation because it is nothing but a compressed knowledge representation. Figure 2.11 illustrates a typical autoencoder.

Figure 2.11
A Typical Autoencoder



Note: Image source: Sarantitis (2020).

A typical autoencoder consists of three parts:

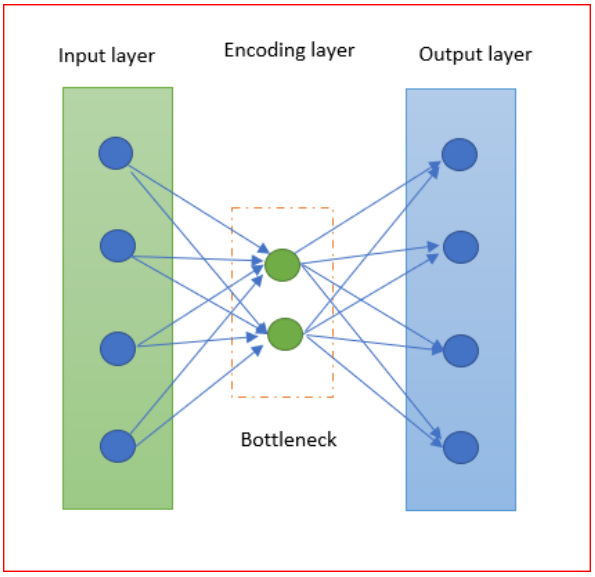
- **Encoder**. In its simplest form, the encoding layer is the first layer, and it consists of as many nodes as there are inputs. With the addition of more layers, the input will be further compressed because of the additional layer. It is complex to figure out how many more layers/hidden layers there should be before the data are passed to latent space.
- **Latent space**. The middle-hidden layer is known as latent space or bottleneck, and it exists between an encoder and a decoder. This layer has a smaller number of nodes than the encoder layer or decoder layer.
- **Decoder**. The final step is to reconstruct the input data at the end of the network. This is where we define how many layers can be added to the model depending

on a number of factors. Thus, it can be said that it helps to decompress the data and then reconstruct the knowledge representation from the original encoded form, and the number of nodes that it has will be the same as the number of input layers.

There's a similarity between the encoder and the decoder, where the encoder is the alter ego of the decoder. An encoder consists of convolutional blocks followed by a group of components that come together to form the input to the autoencoder. The encoder consists of the same number of nodes as the number of features from the dataset. The output of the encoder goes to the compressed layer or latent space.

Despite its importance, the bottleneck is the smallest part of the neural network (Figure 2.12). Latent space is by far the most significant part of the autoencoder, and one can consider it to be the heart of the classifier, considering how much it contributes to the accuracy of the classifier's performance. Information is limited between encoders (inputs) and decoders (outputs) due to the way they are connected. When the input is received, the bottleneck creates a representation of knowledge from it, thereby maximizing the amount of information available from it. Thus, encoder-decoder structures enable the most information to be extracted from images and useful correlations to be established. By compressing the input, the bottleneck prevents the neural network from memorizing or overfitting. It is a good rule of thumb that the smaller the bottleneck, the less chance there is of overfitting.

Figure 2.12
Autoencoder Showing Bottleneck

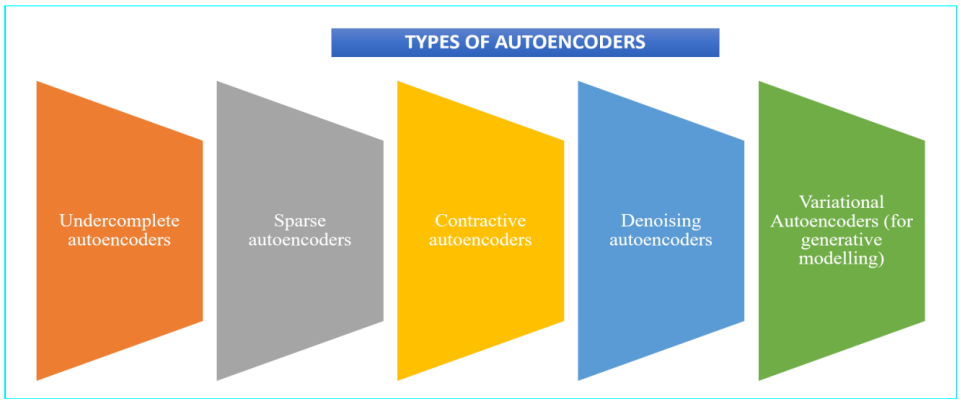


Note: Image source: Bandyopadhyay (2022).

Finally, a number of components that make up a decoder, including up-sampling, preprocessing, and autoencoder, take advantage of the up-sampling of the bottleneck to reconstruct its output. The decoder operates as a “decompressor” of compressed knowledge representations, which can in turn be used to reconstruct the data from its latent attributes to decode it.

It was originally intended that autoencoders would be used to reduce the dimensions of a dataset and to learn features, but in recent years, one of the most common uses for autoencoders has been to learn generative models based on the features that are learned. A number of autoencoders are popular. This section reviews five types (Figure 2.13).

Figure 2.13
Types of Autoencoders



2.6.1. Undercomplete Autoencoders

The main objective of incomplete autoencoders is to capture the most important aspects of data in the most effective and efficient way possible. Compared to the input layer, this layer contains a very small number of layers, as the input layer contains the same number of nodes as the number of features in the dataset. There is no requirement to apply the regularization function to maximize the probability of data, so the probability of data can be maximized without applying the regularization function.

2.6.2. Sparse Autoencoders

Sparse autoencoding is characterized by having a larger number of layers in latent space than input layers of the encoder or input layer. By using this method, more data points can be found in the dataset. In addition, the hidden layer is subjected to a sparsity constraint. Because of this, the input should not be copied to the level of the decoder. In order to improve the efficiency of sparse autoencoding, it is necessary to ignore the hidden layer nodes with the lowest activation values. One of the well-known facts about

autoencoders is that a limited number of hidden nodes can reduce the number of hidden nodes in autoencoders at the same time as the number of hidden nodes can be reduced.

2.6.3. Contractive Autoencoders

The most common use of the contractive autoencoder is when there are only a few variations in the dataset, which is much of the time. To make data representation more robust, loss functions are penalized to improve their robustness. In addition to its use as a tool for the regulation of information, the contractive autoencoder can also be used as a tool for generalization. Basically, contractive autoencoder is a type of autoencoder that generates mappings that tend to contract the data in a strong and predictable manner.

2.6.4. Denoising Autoencoders

Denoising autoencoders intentionally apply noise to the dataset, which results in corrupted output datasets. As part of this process of introducing noise, a random substitution of zeros with the original data is used to introduce noise in a systematic way. A denoising autoencoder's function is to remove the noise from the original dataset. In order to do this successfully, it needs to align the data with the original dataset. In this process, the loss function must be minimized until convergence is achieved by comparing the output with the input. The latent representation of the data is derived based on the coding of the data using autoencoders.

2.6.5. Variational Autoencoders

The variational autoencoder is based on Bayesian machine learning and DNN, and it aims to tackle variational inference due to its Bayesian approach. As a result, it is possible, first of all, to reconstruct the input and, second, to learn efficient representations

of data through the bottleneck in the process. The data generated by a directed graphical model has a few important differences from other autoencoders, since variational autoencoders are Bayesian autoencoders. Because of this, each layer of interest is represented as a distribution.

2.7. Conclusion

An extensive review of journal articles, conference papers, and books was conducted. Based on this literature, financial companies continue to face huge problems with credit card fraud. There are a few methods of detecting credit fraud using deep learning. To solve the problem of credit card fraud, this study introduced the newly mature deep learning and autoencoders method, as detailed in the next chapter.

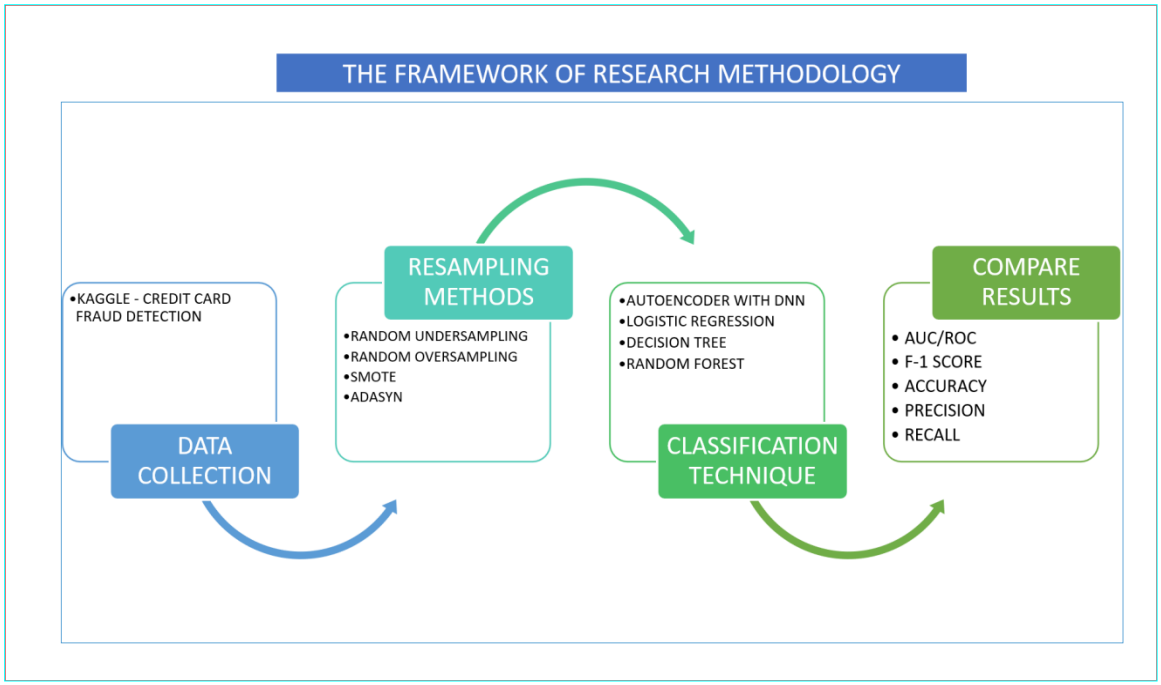
CHAPTER 3:

METHODOLOGY

3.1. Introduction

The purpose of this chapter is to discuss the method used in this study to differentiate between nonfraudulent transactions and fraudulent transactions. Figure 3.1 illustrates the end-to-end process of detecting credit fraud using a deep neural network (DNN) and autoencoder.

Figure 3.1
The Framework of Research Methodology



As a part of this process, multiple steps of data processing were undertaken, including data cleansing and preprocessing. This chapter begins with a discussion of data collection, data preprocessing, data standardization, and different resampling methods, including undersampling and oversampling techniques like synthetic minority

oversampling technique (SMOTE) and adaptive synthetic (ADASYN). Once the proper sampling method has been selected, different classification techniques are used to determine whether a given transaction is fraudulent. The last step in the model development process is evaluation of the classifiers. A large variety of metrics are available—such as F1 score, receiver operating characteristic (ROC), accuracy, precision, and recall—to evaluate the performance of the model. As soon as we understand the model's performance, we will be able to predict its performance on unseen data.

3.2. Data Collection

This study used an anonymized dataset obtained from Kaggle containing credit card transactions made by European cardholders. Data consisted of 284,807 credit card transactions over the course of 2 days in September 2013. There were no missing data in the dataset, and only 492 out of 284,807 transactions were fraudulent, making it heavily skewed. The dataset contained 30 features, of which only two could be determined, namely the amount and the time. To maintain anonymity, principal component analysis (PCA) was applied to the remaining 28 features. All transactions were already labeled as fraudulent or not before they were released. Table 3.1 lists the variables, Table 3.2 provides a sample, and Table 3.3 provides descriptive statistics for the dataset.

Table 3.1
Variables in the Kaggle Dataset

Features	Description
Time	Time elapsed between transaction
V1…V28	Through PCA, sensitive characteristics are reduced to non-descriptive variables.
Amount	Amount(Tx)
Class	1 = Fraud; 0 = Non-Fraud

Note: The dataset included 284,807 observations and 31 features.

Table 3.2
Credit Card Transaction Sample Dataset

Sr No	Time	V1	V2	….	V27	V28	Amount	Class
0	0	-1.35981	-0.07278	….	0.133558	-0.02105	149.62	0
1	0	1.191857	0.266151	….	-0.00898	0.014724	2.69	0
2	1	-1.35835	-1.34016	….	-0.05535	-0.05975	378.66	0
3	1	-0.96627	-0.18523	….	0.062723	0.061458	123.5	0
4	2	-1.15823	0.877737	….	0.219422	0.215153	69.99	0

Table 3.3
Summary of the Dataset

	Count	Mean	SD	Min	25%	50%	75%	Max
Time	69854	34638.55	14403.38	0	28735.5	37884	45421	53609
V1	69854	-0.24488	1.866654	-56.4075	-1.00564	-0.24343	1.153672	1.960497
V2	69854	-0.02065	1.654683	-72.7157	-0.59234	0.071851	0.727702	18.90245
V3	69854	0.681629	1.42028	-33.681	0.195094	0.77086	1.403483	4.226108
V4	69854	0.169481	1.37528	-5.1726	-0.72419	0.188002	1.052453	16.71554
V5	69854	-0.2714	1.393637	-42.1479	-0.88909	-0.30343	0.267663	34.80167
V6	69854	0.101208	1.307236	-26.1605	-0.63644	-0.15055	0.494044	22.5293
V7	69854	-0.11522	1.259963	-31.7649	-0.60455	-0.0747	0.419725	36.67727
V8	69854	0.057927	1.236014	-73.2167	-0.14123	0.06685	0.345317	20.00721
V9	69854	0.02074	1.162287	-9.28393	-0.67161	-0.06992	0.673668	10.39289
V10	69854	-0.03861	1.092254	-18.2712	-0.51193	-0.09827	0.436675	13.19823
V11	69854	0.302028	1.091218	-4.0499	-0.4913	0.242751	1.108539	12.01891
V12	69854	-0.18456	1.222588	-17.7691	-0.64477	0.054765	0.590253	7.848392
V13	69854	0.074523	1.079426	-5.79188	-0.64384	0.041248	0.771152	4.465413
V14	69854	0.104817	1.037181	-19.2143	-0.32188	0.101633	0.55234	10.52677
V15	69854	0.155092	0.938081	-4.15253	-0.39919	0.261368	0.831284	5.784514
V16	69854	-0.00861	0.923532	-13.5633	-0.48997	0.061768	0.546953	6.098529
V17	69854	0.085179	1.017893	-25.1628	-0.38277	0.027176	0.495759	9.253526
V18	69854	-0.09389	0.86071	-9.49875	-0.56945	-0.08289	0.397427	5.041069
V19	69854	-0.02337	0.817597	-7.21353	-0.51772	-0.02859	0.484071	5.228342
V20	69854	0.046863	0.751094	-15.8065	-0.16747	-0.02374	0.171509	39.4209
V21	69854	-0.02847	0.73929	-34.8304	-0.2259	-0.06078	0.115849	22.61489
V22	69854	-0.10635	0.638581	-10.9331	-0.52666	-0.08069	0.309358	10.50309
V23	69853	-0.03853	0.610415	-26.7511	-0.17901	-0.05134	0.079195	17.29785
V24	69853	0.005991	0.59644	-2.83663	-0.32566	0.062094	0.403217	4.014444
V25	69853	0.13616	0.439287	-7.49574	-0.12758	0.174248	0.422311	5.525093
V26	69853	0.022047	0.497722	-2.53433	-0.32875	-0.07509	0.299552	3.517346
V27	69853	0.003067	0.381777	-8.56764	-0.06246	0.009509	0.082528	11.13574
V28	69853	0.003218	0.321419	-9.61792	-0.00582	0.022783	0.075841	33.84781
Amount	69853	96.99289	271.2728	0	7.65	26.5	88.72	19656.53
Class	69853	0.002491	0.049848	0	0	0	0	1

3.3. Data Preprocessing

It is essential to preprocess data before implementing a machine learning algorithm, since each model produces a different specification for the predictors, and data training can affect the results. Data preprocessing focuses on making data more concise, more variable, and containing fewer missing values by cleaning and preparing it. Three main steps are involved in preprocessing:

- Data formatting. Successful machine learning requires a specific format, which is why datasets must be formatted in such a way as to ensure that the algorithms can make accurate predictions or models based on the data. For this study, comma-separated files containing transaction records were used.
- Data cleansing. The purpose of data cleaning is to format the data, fix inaccurate data, and identify duplicate records. There are some columns that are missing values, and irrelevant records need to be removed.
- Data sampling. Samples are taken from a large population of observations and then analyzed, which is the process of statistical analysis. As a result, the model is trained more effectively, and a better understand can be obtained of the pattern, behavior, and other characteristics of the dataset.

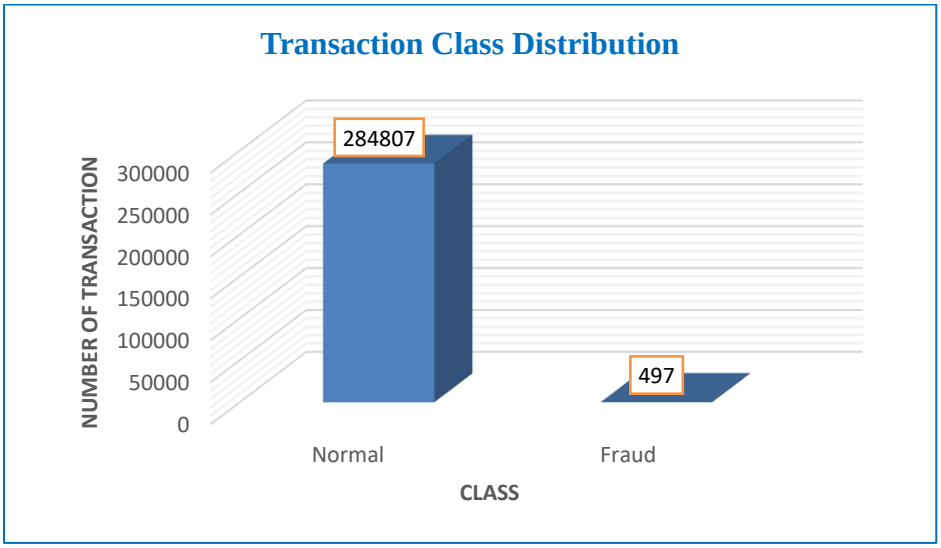
The remainder of this section reports on aspects of the data preprocessing related to checking for unbalanced data, features correlation, and feature density plot and ensuring data standardization.

3.3.1. Imbalanced Data

Generally, a dataset is referred to as imbalanced if one of the class labels has a very high number of observations, while the other has a very low number of observations.

As an example, for credit card fraud, the number of normal transactions is much higher than the number of fraud transactions. As shown in Figure 3.2, only 492 (or 0.172%) of transactions in the study dataset were fraudulent.

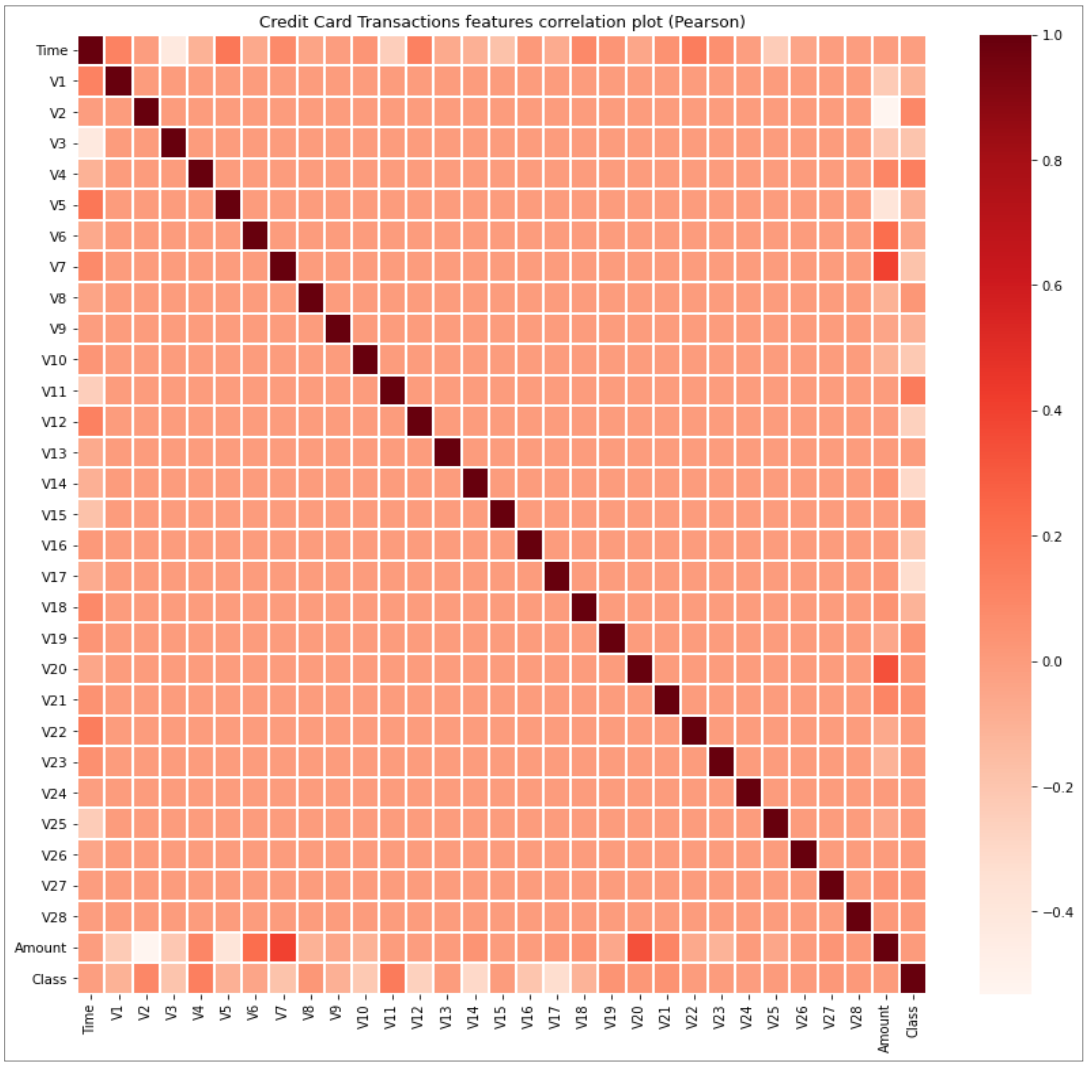
Figure 3.2
Distribution of Class



3.3.2. Features Correlation

A heatmap can be described as a visual representation in which the values of each individual data point are represented by colors. Using this method, numerical data can be represented graphically with respect to a number. Since the data points are from PCA, they were uncorrelated. The histogram in Figure 3.3 indicates no correlation between the features from V1 to V28.

Figure 3.3
Feature Correlation

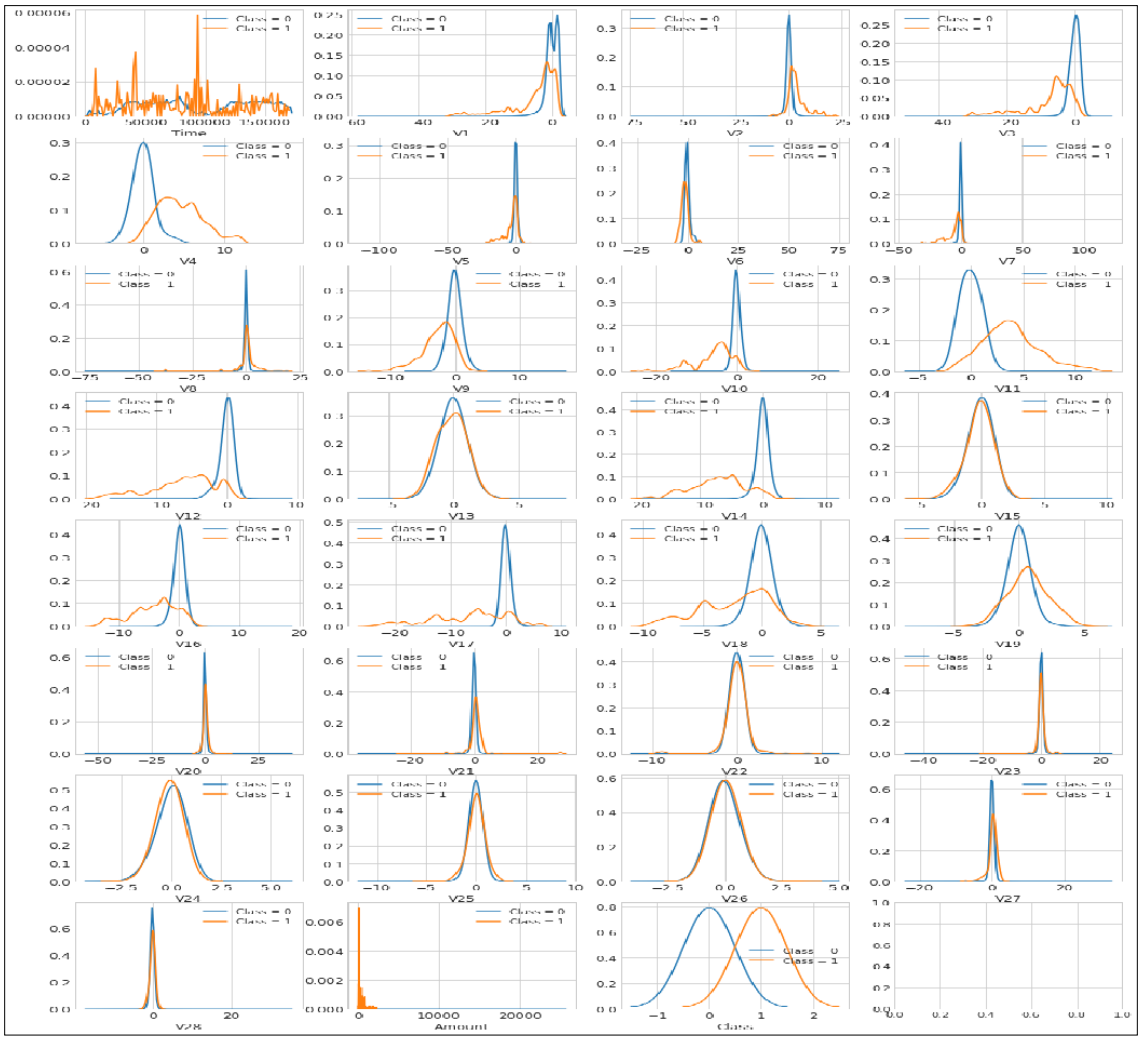


3.3.3. Feature Density Plot

A feature density plot is useful for analyzing the distribution of numeric variables within a group. By using density plots, it is possible to determine where the values are concentrated and where the peaks are located. In this study of attribute distributions, density plots were used. It was necessary to check the distribution of attributes individually to provide high-quality information.

There was a clear distribution of values between 0 and 1 for the class or target variables, as seen in features V4 and V11 (Figure 3.4). In addition, the profiles of V3 and V10 were quite different from one another. Because of this, both V26 and V28 had similar profiles for the two types of class values, whereas the two types of class values for V26 and V28 did not have similar profiles for the two types of class values. There are similarities between the correlation plot and the density plot when we examine both of them, for instance, V20 is having higher values, which is one of the significant characteristics of the correlation plot.

Figure 3.4
Feature Density Plot



3.3.4. Data Standardization

The standardization of data is one of the most important steps in the preprocessing of data. Data are realigned using the properties of a normal distribution (mean = 0 and standard deviation = 1), which allows the data to be converted into a uniform format. Many machine learning models require that the features be standardized. For Kaggle data, the Python package sklearn was used to standardize the amount feature. Standardization can be achieved as follows, where.

$$z = \frac{x - \mu}{\sigma}$$

$$\text{Mean} = \frac{1}{N}\,\Sigma x_i$$

$$\text{Standard Deviation} = \sqrt{\frac{1}{N}\sum_{i=0}^{n}(x_i - \sigma)^2} \qquad 3.1$$

3.4. Class Imbalance and Data Resampling

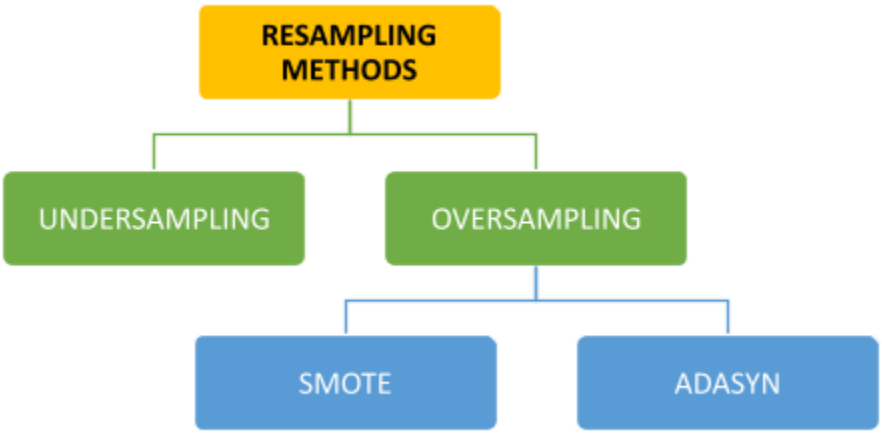
In highly imbalanced datasets, machine learning models tend to pick the most popular patterns and ignore the rest. In this dataset, 99.99% of the data was labeled ‘not fraud’ and remaining was labeled ‘fraud.’ Even if a model correctly classifies everything as not fraud, it will be 99.9% accurate. Is the model effective? No, because no transaction is considered fraud. Hence, no matter how accurate the model is, it is useless!

It is necessary to use some strategies in such a dataset or to use other metrics (apart from accuracy) in such scenarios. In the case of machine learning, excellent results

can be achieved when it is trained on a balanced dataset. This means that two classes should have an equal number of samples within them. This is because machine learning gives the most accurate results because the algorithm is designed in such a way that it minimizes error levels. This improves the accuracy of the machine learning model.

The trained model will be more likely to favor the majority class when the dataset is highly imbalanced. The results that have been obtained because of this will be of no use for the future. Various types of resampling techniques are available for resampling of data (Figure 3.5).

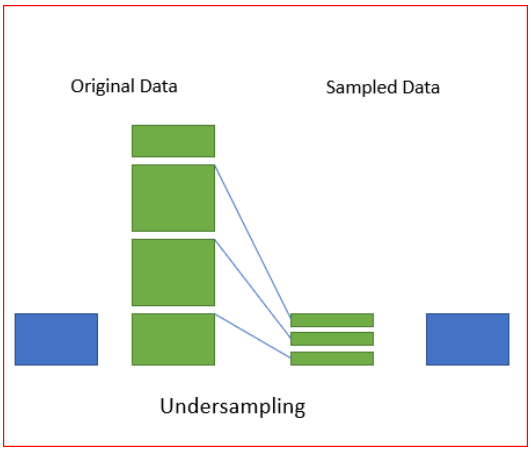
Figure 3.5
Resampling Methods



3.4.1. Undersampling the Negative Class

In this method, the majority class of legal transactions is undersampled so that a greater proportion of fraudulent transactions can be identified (Figure 3.6).

Figure 3.6
Illustration of Undersampling



Note: Image source: Al-Rahman Al-Serw (2021).

Undersampling techniques are commonly used to ensure that classification algorithms can train on the balanced dataset so that they can make accurate predictions. In this technique, majority class samples are removed in order to equalize the number of minority class samples with the number of majority class samples. It is possible for rule-based classifiers to lose useful samples when this process occurs, resulting in data inefficiency. Although this technique is effective in spite of severe imbalances, it is only useful when minority classes have a sufficient amount of data.

There are many advantages of undersampling, including that it reduces the chance of bias in classifiers and that it prevents overfitting in the classification models. In the absence of resampling, it might be possible to run a classification model with 90% accuracy. Closer inspection may show that the results are heavily skewed towards the majority.

In addition, undersampling has some major disadvantages. One of the major disadvantages is that when there is not enough information, machine learning is not able

to learn the pattern and cannot make a generalization, which may lead to a false prediction. An undersample can affect the majority class by removing all relevant data points.

Undersampling leads to loss of much information that may be important for the classifier. In conclusion, if a sufficient amount of majority class data is removed, the remainder of the dataset will not have enough data for the rule to be formed, and the accuracy of the classifier will drop.

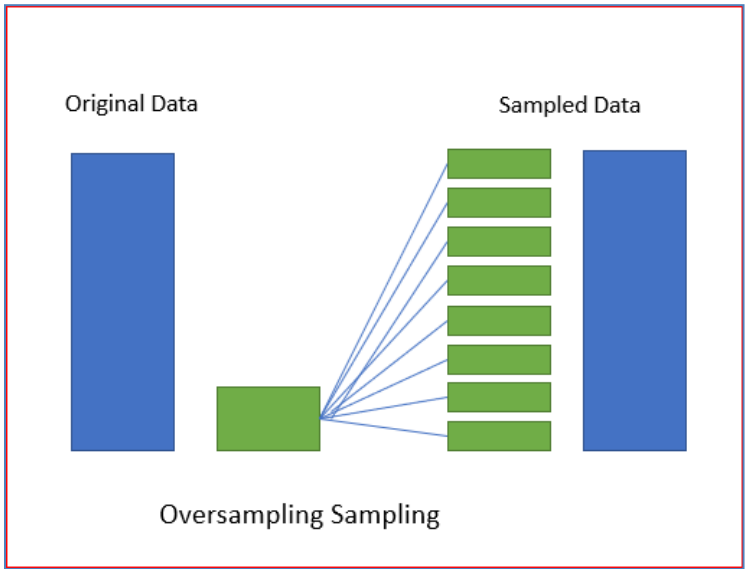
This study applied a 70:30 split to the dataset, which means that 70% of the data was used for training, while 30% was utilized for testing purposes. Undersampling the training data is used to match the minority class with the majority class. A total of 692 samples were obtained after undersampling to train the model.

Due to the problems associated with the undersampling technique, an oversampling technique was used in this study. The main advantage of oversampling is that it is more effective, and there are several oversampling techniques available, including SMOTE and ADASYN.

3.4.2. Oversampling the Positive Class

Oversampling is the polar opposite of undersampling. Minority class samples are duplicated randomly to align them to the percentage of samples in the majority class (Figure 3.7). Overfitting occurs because the minority class samples are duplicated exactly (there is no added information to the model). Two methods of oversampling are SMOTE and ADASYN.

Figure 3.7
Illustration of Oversampling

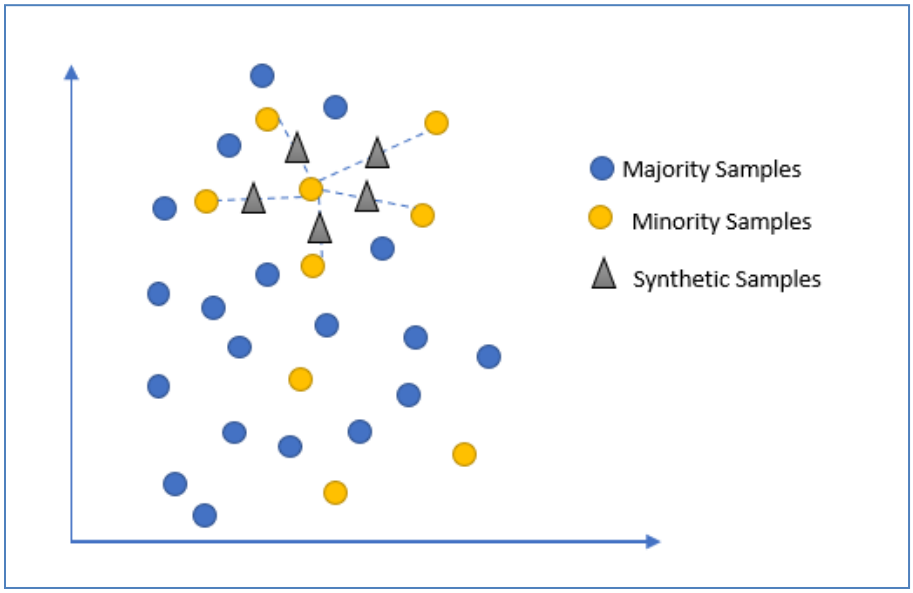


Note: Image source: Al-Rahman Al-Serw (2021).

3.4.2.1. Synthetic Minority Oversampling Technique

A simple solution to the problem is to duplicate samples from the minority class. SMOTE was proposed by Nitesh Chawla and his team to generate synthetic samples. There will be K nearest minority classes to the observation, and these samples will be generated randomly. As shown in Figure 3.8, SMOTE uses the K nearest neighbors where k = 5; this means that, in order to create one sample, samples from five neighbors are considered. One of the biggest disadvantages of SMOTE is that it does not take advantage of the distribution of the target object, so there is a possibility of overlapping, which will introduce additional noise into the analysis.

Figure 3.8
SMOTE



Before SMOTE was applied, the dataset was divided into two parts in a 70:30 ratio, i.e., a training set and a testing set. In this case, the minority class was used to generate the synthetic using SMOTE. SMOTE oversampling resulted in 395,058 samples in the training set and 85,443 samples in the testing set.

3.4.2.2. Adaptive Synthetic Sampling Approach

According to He et al. (2008), ADASYN produces synthetic samples for minority classes, similar to SMOTE. This method relies on generating more synthetic data for observations that are harder to learn than for observations that are easier to learn. Further, the algorithm uses the K nearest neighbors' method to get the opinions of the minority classes within the region of K nearest neighbors. However, this method results in a more significant representation of minority classes within the region. Furthermore, if the majority of observations cannot be found within the range of K nearest observations, then

no sample will be created. Additionally, ADASYN considers density distribution; based on this distribution, samples are created, which allows the decision boundaries to be adjusted. It has been found that ADASYN offers a number of advantages, not the least of which is its adaptive nature. This generates more data for examples that are more difficult to learn, as well as the ability to sample more negative data.

After dividing the training data with a 70:30 split and applying ADASYN oversampling to the minority class, 398,048 samples were generated. A model was trained using 70% of the data and was tested using 30% of the data, i.e., 85,443 samples in total.

3.4.3. Comparison Between Random Undersampling, Random Oversampling, SMOTE, and ADASYN

Table 3.4 compares process, strengths, and limitations for the resampling methods.

Table 3.4
Comparison of Different Sampling Techniques

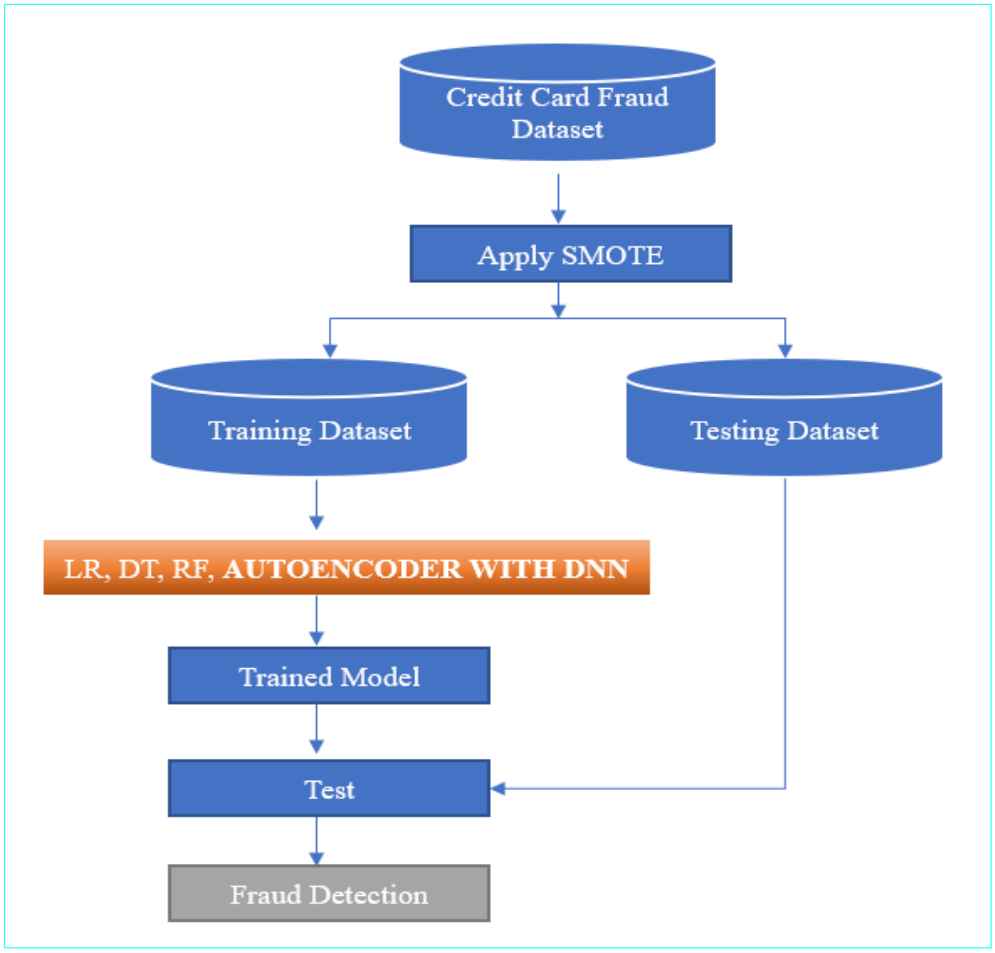
	Random undersampling	Random oversampling	SMOTE	ADASYS
Process	Reduces majority class samples by randomly discarding it	Increases the minority class mostly by duplicating records	Oversamples by creating synthetic records	Works the same as SMOTE and generates synthetic samples
Strength	Efficiencies in convergence	No information loss	Increase in overall effectiveness	Improvement in the overall performance of the model
Limitations	Loss of information from the features	Faces overfitting problem	Generates noise in the dataset	May affect the precision and recall of certain models

Note: SMOTE indicates Synthetic minority oversampling technique; ADASYS, Adaptive synthetic.

3.5. Model Development

Several techniques can be used to be able to detect fraud in credit card transactions. The objective of this study was to compare the performance of several machine learning algorithms in order to identify which one performs the best in detecting fraud from credit card transactions. These algorithms included a logistic regression analysis, a decision tree analysis, a random forest analysis, and a hybrid model that combined an unsupervised learning technique called autoencoder with a DNN to detect fraud in credit card transactions (Figure 3.9).

Figure 3.9
Credit Card Fraud Detection Framework



3.5.1. Experiment Setup

This praxis used Google Colab as its environment and Python as its language of choice. Several libraries and frameworks are available in Python for the implementation of deep learning algorithms. As part of Google's Colab notebook, deep learning algorithms can be run on tensor processing units and graphics processing units. Using Pandas, a complete set of libraries was imported and then the Kaggle dataset was loaded.

3.5.2. Training and Testing Dataset

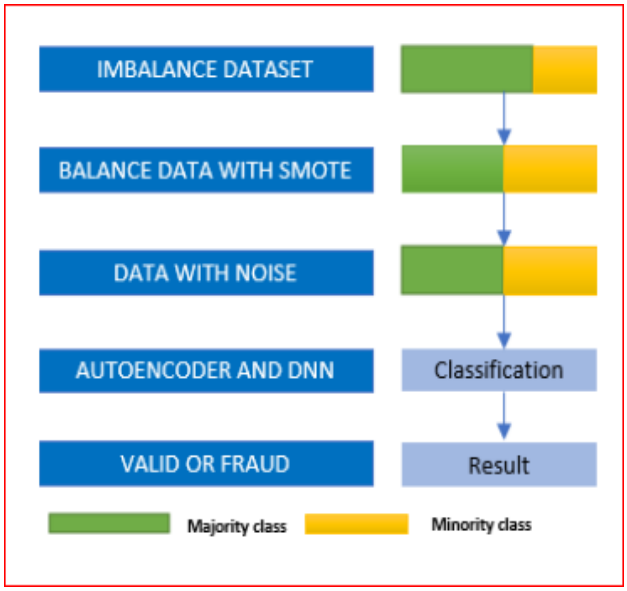
In the machine learning process, the dataset set is divided into training and testing sets (validation sets) before any classifiers are applied to the dataset set. The first step before starting to train the model was to divide the dataset into sections to simplify the training process. As part of the process of building the model, we use training data, and as part of the process of validating the dataset, we use testing data. For this dataset, 70% was used for training purposes, while 30% was used for validation/testing of the system, as part of the verification process.

3.5.3. Proposed Model: Autoencoder with Deep Neural Network

In the development of a machine learning model for classification data using imbalanced datasets, several serious issues arise related to accuracy. As a result of oversampling, a new sample is synthesized for the minority class to balance the sample between the majority class and the minority class, but there is a risk that noise will be introduced into the data. There is still a problem with noise being introduced into the sampled dataset after applying the oversampling method. To solve the problem of noise,

the autoencoder unsupervised deep learning method was used to remove the noise from the sampled dataset. This step is illustrated in Figure 3.10.

Figure 3.10
Proposed Model Using Autoencoder and Deep Learning



Generally, the idea was to first oversample the minority class using SMOTE. Then, an unsupervised learning method from a deep learning autoencoder was used to remove noise from the sample data. After the noise was removed, we used this dataset to train a DNN. To understand the performance of the model, we used the test dataset after training to calculate metrics to evaluate the model's performance.

3.5.3.1. Denoising Oversampling Data

The previous section considered several sampling techniques to balance the majority class and minority class. Resampled data was passed through the autoencoder to denoise the data. Figure 3.11 shows the design of the autoencoder to remove the noise, and Figure 3.12 illustrates the denoising process using the autoencoder.

Figure 3.11
Autoencoder

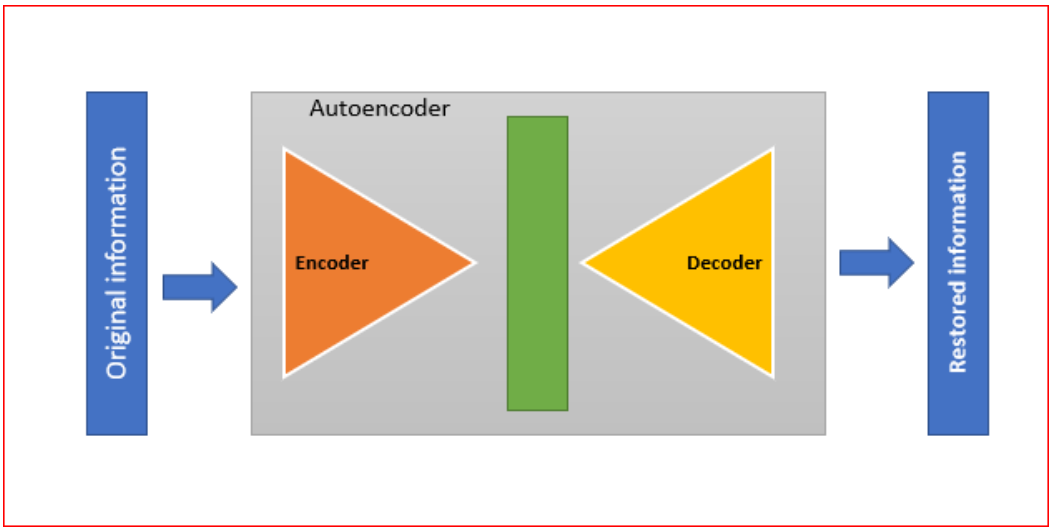
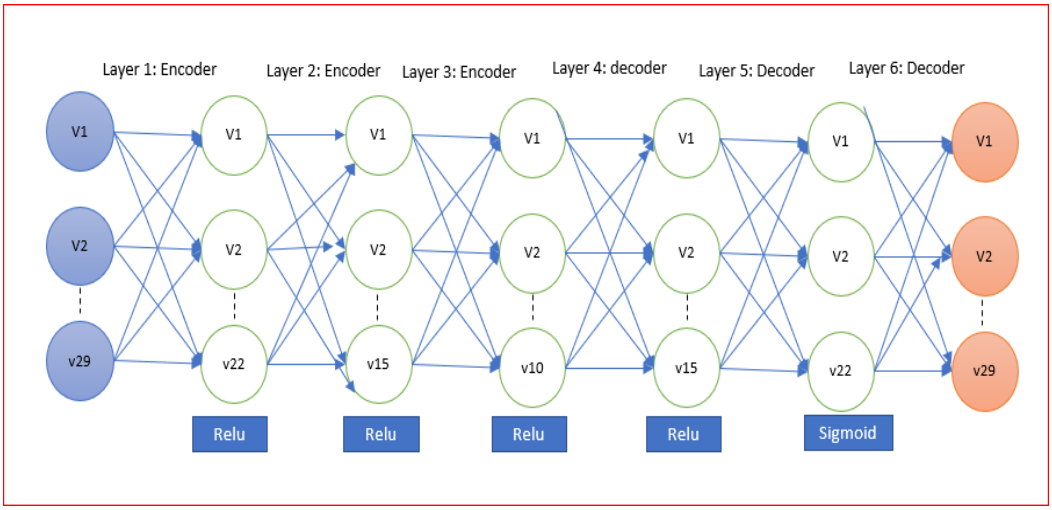


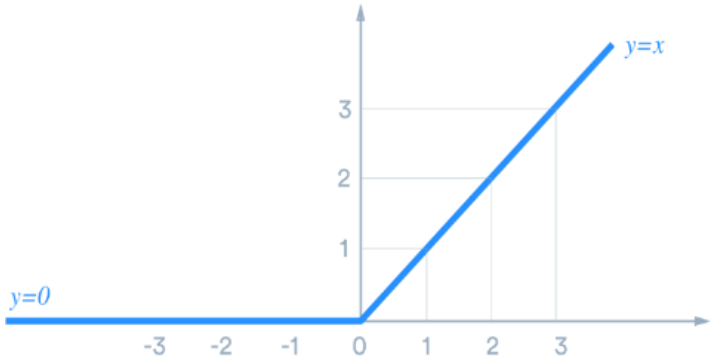
Figure 3.12
Denoising Oversampling Data



In its most basic form, an autoencoder is a neural network that automatically learns how to encode and decode. As a result, the encoder encrypts the data into a lower dimension representation before decoding it back to its original format. In terms of design, it is composed of three encoding layers and three decoding layers. Relu was used

as the activation layer in the hidden layer and sigmoid was used at the output layer. The equation is y = max(0, x). The range of this activation function is (0, inf) (Figure 3.13).

Figure 3.13
Relu Function

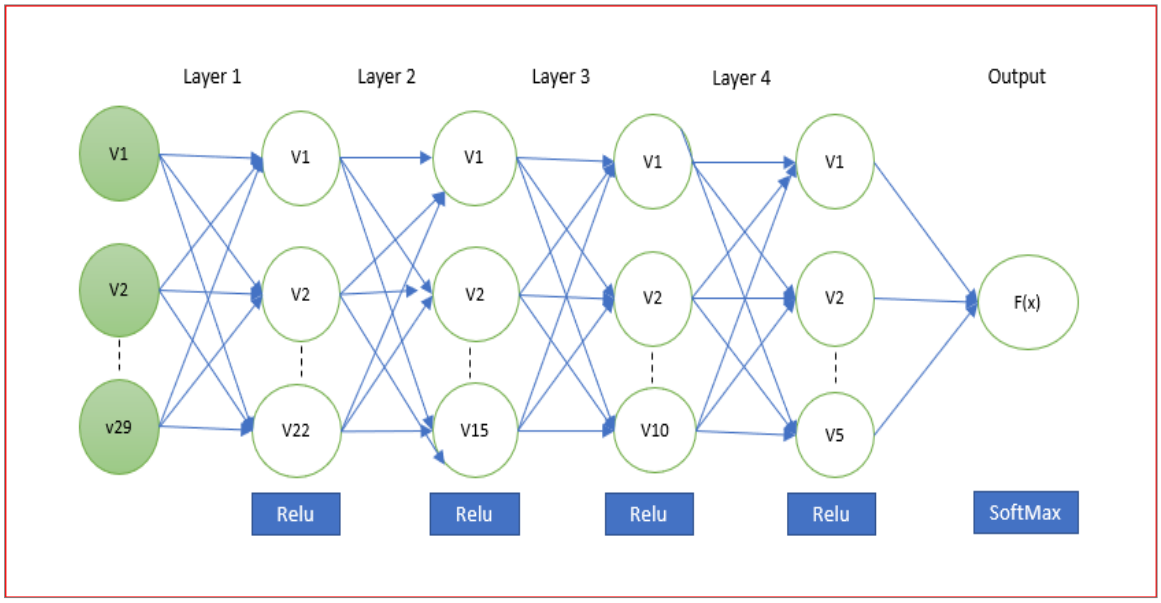


As a first step, all the necessary libraries were loaded. These were standard libraries available in the Python environment. NumPy and Pandas were imported for matrix calculations and for reading the .csv files. To visualize the data, Matplotlib and Seaborn libraries were used. Following this, the Keras framework was imported from the sequential framework in order to be able to create several dense layers based on the design of the autoencoders.

3.5.3.2. Classification Using Deep Neural Network

The denoised dataset received by the autoencoders was the output in the previous step. With the help of this dataset, a DNN was trained. The neural network model design is shown in Figure 3.14. Essentially, it was composed of five interconnected layers.

Figure 3.14
Classification Using Neural Network



Input layers consisted of 29 neurons, Layer 1 consisted of 22 neurons, and so on. The activation function at the hidden layer is Relu, and at the output the activation function is SoftMax. As a result of the classification process, the output layer was where the classified output was obtained. An output of 1 indicated fraud and 0 indicated normal behavior.

3.5.4. Machine Learning Classification Techniques

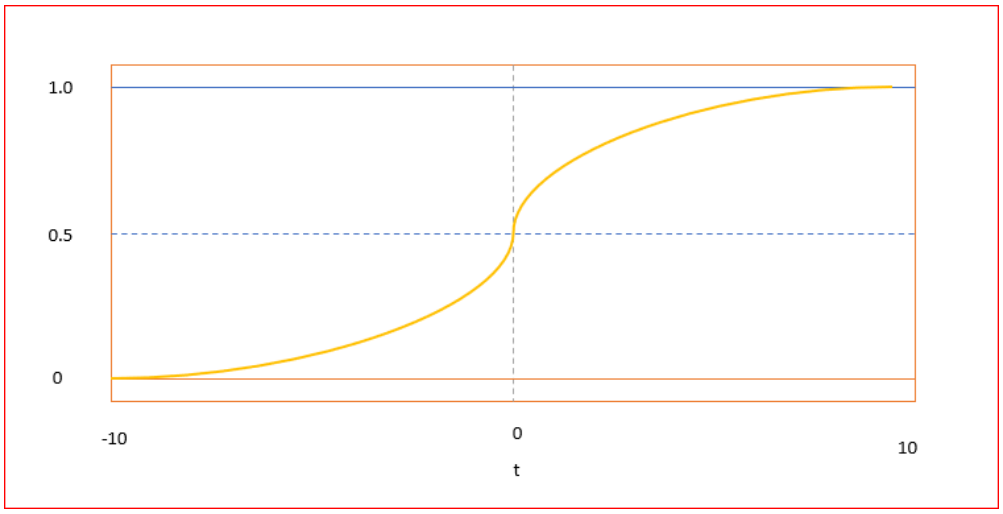
The credit card dataset had a binary classification of nonfraudulent (0) or fraudulent (1). The methods for resampling the data had to be evaluated by training the models using classifiers with resampled data to determine whether they were accurate. In this study, three different classification techniques were examined: logistic regression, random forest, and decision tree.

3.5.4.1. Model 1: Logistic Regression

Logistic regression calculates the weighted sum of input features and biases. In logistic regression, the threshold value is used to find the probability of either 0 or 1. It is generally accepted that data that exceed or equals a threshold value are rounded up to 1, while data beneath the threshold value are rounded down to zero (Equation 3.2, Figure 3.15).

$$\hat{P} = H_\theta\ (X) = \sigma(\theta^T X) \qquad 3.2$$

Figure 3.15
Logistic Regression

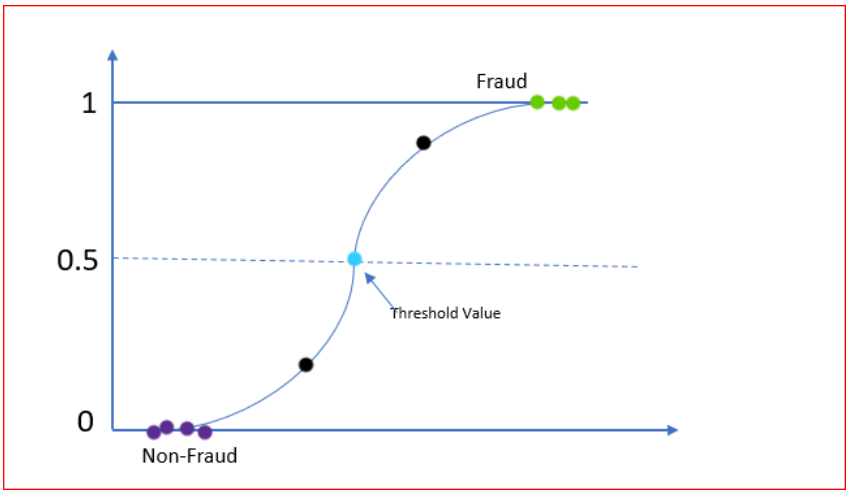


Note : Image source: www.oreilly.com.

It is possible to derive an equation for logistic regression from a linear regression equation, and the other way around.

As a result, the fraud class was assigned a value of "1" and the nonfraud class was assigned a value of "0". As shown in Figure 3.16, using a threshold of 0.5, it was possible to differentiate between the two classes.

Figure 3.16
Logistic Regression: Fraud vs Non-Fraud



Note: Image source: Alenzi and Aljehane (2020).

In the previous paragraphs, it was mentioned that the dataset consisted of two parts, a training set and a testing set. Approximately 70% of the dataset was used for training, and about 30% of the dataset was used for the testing.

3.5.4.2. Model 2: Decision Tree

The use of decision trees has proven to be one of the most effective methods for classifying data. Fast and reliable, the method is known for its transparency. Machine learning algorithms such as classification and regression trees (CART) are extremely popular. A decision tree partitions the data space into smaller subspaces by assigning labels to each subspace. To train trees, each node is split separately, with the algorithm trying to maximize the split along all axes. By analyzing several metrics, such as entropy, information gain, and Gini, it can be determined which split point results in the least impurity. After the tree is built, it can be traversed based on the rules to determine the predicted label and its probability for new records.

As a result of overfitting, overly complex trees are created that can't generalize to unknown data. There are several ways to overcome overfitting. When training trees, it is important to control their growth, so they do not grow too deep or split too many leaf nodes at the same time. A tree can also be pruned after construction. Using minimally complex pruning techniques, some subtrees are selected to prevent overfitting.

$$\text{Gini impurity: } I_g(p) = \Sigma p_i(1 - P_i) = 1 - \Sigma P_i^2$$

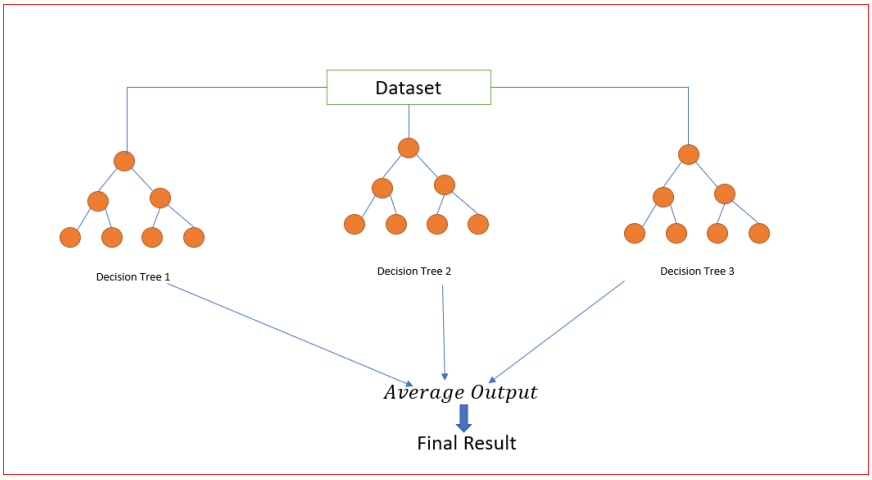
$$\text{Information gain : } H(_p) = -\sum_1^c p_i \log_2 p_i \qquad 3.3$$

Pi represents the probability that an observation of class *i* will be found. First, the data needed to be shuffled and split into two parts—a training set and a test set, with 70% of data going to training. Oversampled data was used for training the model.

3.5.4.3. Model 3: Random Forest

Random forest is among the most widely used ensemble techniques. Although this algorithm can be used to solve problems related to regression and classification, it is mainly used to solve problems related to classification. Random forest algorithms can construct decision trees using data samples. In addition, the algorithm depends on the quality of the discrete trees and the correlation between them. As a result, a large number of considerations can be incorporated into prediction by distinguishing various information factors. A small amount of data is sufficient for the algorithm to work. Aggregating uncorrelated trees is the main concept that makes the random forest algorithm superior to decision trees. To create a better random forest, we need to create several model trees and make an average of them (Figure 3.17).

Figure 3.17
Random Forest



Note: Image source: TIBCO (n.d.).

SMOTE in the previous section oversampled with a target of 1:10 (total of 10%), which essentially created new entries for the smaller class (frauds) by matching the original property values. With the random undersampler method, some entries were selected from the biggest class (valid) at random and removed from the dataset based on a proportion of 5:10 (double the minority class). As a result, a third of the whole set was classified as fraud. This transformation was applied only to the training set. To properly evaluate our model, the testing data (30% of the full dataset) needed to be as real as possible.

In this part, we chose the random forest classifier from the Scikit-Learn Python library after resampling. A total of 100 estimators (decision trees) were used with the standard hyperparameters from the model (only setting a random_state for reproducibility). The dataset was divided into training and testing data, as well as independent values (X) and target values (Y).

Random forest has several advantages. As the random forest algorithm depends on multiple trees that are trained individually based on the data, it is not biased and has a reduced amount of bias overall. This is because each tree is trained separately based on the data. The algorithm is very stable. The introduction of a new data point in a dataset does not affect the overall algorithm but affects only a single tree.

In addition, random forest algorithms generally perform better than single decision trees. As a result, they are less prone to overfitting and more robust to noisy data, and they can handle high dimensionality quite well. Random forest training and predicting can be highly parallelized since each tree is built independently, speeding up computationally intensive model training. Furthermore, random forest can be applied to solve classification and regression problems.

3.6. Summary

This chapter discussed data preprocessing and different sampling techniques, including random undersampling, random oversampling, SMOTE, and ADASYN, as well as their integration into different classifiers. In the end, it was possible to determine which SMOTE sampling worked best with most classifiers. The proposed model was designed with a variational autoencoder and a DNN to detect credit card fraud using the SMOTE oversampling technique.

CHAPTER 4:

RESULTS

4.1. Introduction

This chapter presents the results of the study with the machine learning models discussed in the previous chapter. The performance of the model was evaluated with an area under the curve (AUC) score and other metrics. This was done to determine its accuracy. Three types of datasets were used for the evaluation: the original dataset, an undersampling dataset, and an oversampling dataset. The goal was to compare the models and determine which was capable of accurately predicting credit card fraud with the greatest probability.

4.2. Performance Measure

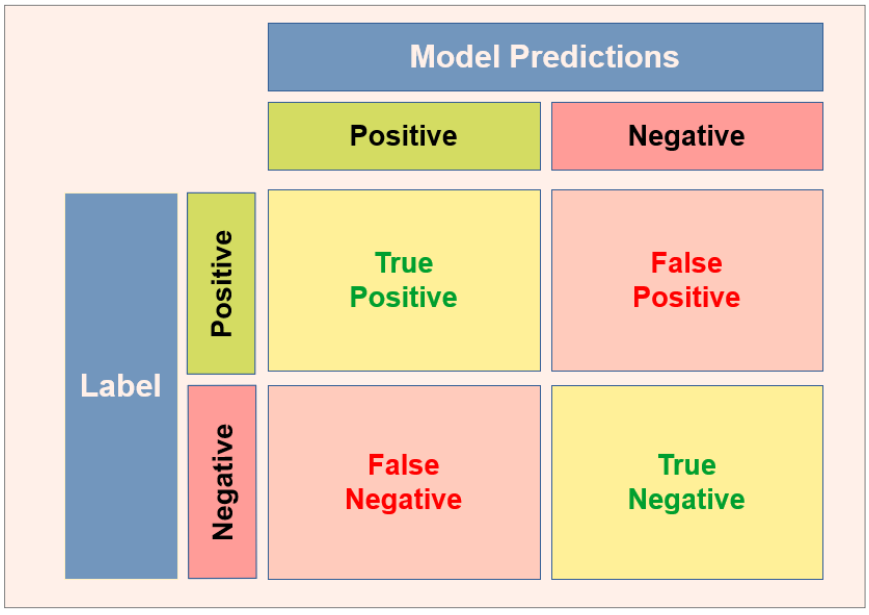
During the development of the model, the performance of the classifier is one of the most critical factors. Before determining the most relevant metrics that can be used to measure the performance of the model, it is critical to consider some key factors. In classification models, numerous metrics can be used, including accuracy, precision, and recall. Since credit card fraud data is highly imbalanced, accuracy is not a good measure since it would be biased towards the majority class. Other measures like F1 score and receiver operating characteristic (ROC) AUC score are very effective for classification models. In addition, it is also relevant to look at the recall and precision of the test as good measures for this type of problem. The sections below define each performance measure.

4.2.1. Confusion Matrix

In classification algorithms, confusion matrices are used to define performance. By using a confusion matrix, it is possible to determine how well an algorithm performs. In a correlation matrix, each cell represents the correlation between two variables. Correlation coefficients are high when there is a strong correlation between two variables. Some performance measures can be calculated from the confusion matrix, such as accuracy, precision, recall, and ROC AUC score. A correlation matrix shows us which pairs of points are the most correlated.

As shown in Figure 4.1, comparing true class with predicted class, results can fall into four categories: true positive (TP), where fraud is correctly identified as fraud; true negative (TN), where nonfraud is correctly identified as nonfraud; false positive (FP), where fraud is incorrectly identified as nonfraud; and false negative (FN), where nonfraud is incorrectly identified as fraud.

Figure 4.1
Confusion Metric



4.2.2. Accuracy

Accuracy is defined as " the total number of correct predictions divided by the total number of predictions made for a dataset" (Brownlee, 2020). The following formula is used:

$$(TP+TN)/(TP+TN+FP+FN) \quad 4.1$$

4.2.3. Precision

"Precision quantifies the number of positive class predictions that actually belong to the positive class" (Brownlee, 2020). To calculate precision, the following formula is used:

$$TP/(TP+FP) \quad 4.2$$

4.2.4. Recall

"Recall quantifies the number of positive class predictions made out of all positive examples in the dataset" (Brownlee, 2020). Recall is calculated with the following formula:

$$TP/(TP+FN) \quad 4.3$$

4.2.5. F1 Score

The F1 score is the "harmonic mean of the two fractions [of precision and recall]" (Brownlee, 2020). F1 is calculated with the following formula:

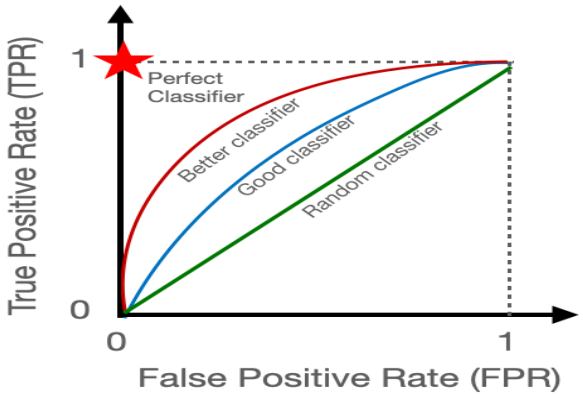
$$F_1\text{–score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2\text{TP}}{2\text{TP} + \text{FP} + \text{FN}} \quad 4.4$$

The ideal is to have model with a precision of 1 and a recall of 1. The F1 score would be 1, i.e., 100% accuracy, which is not always the case for machine learning models.

4.2.6. AUC-ROC Curve

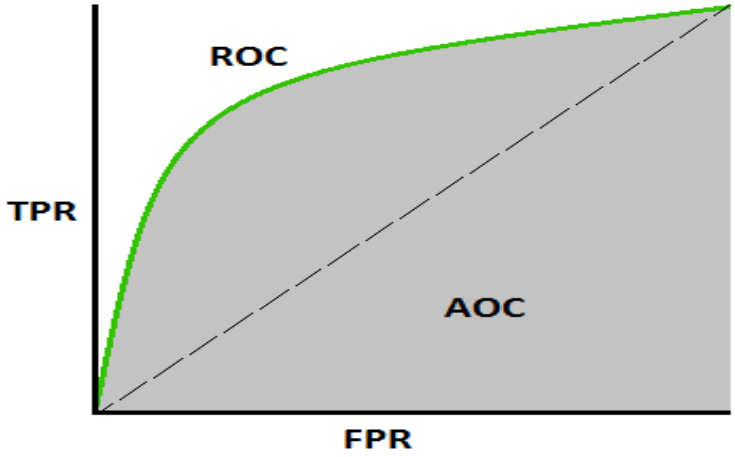
To measure performance in credit card fraud detection classification problems, the ROC will be used. It is basically a ratio between TP and FP. This AUC-ROC is plotted at different threshold settings. An AUC can be used as a measure of how effectively the model can distinguish between two classes. For this purpose, the probability curve can be used, which basically reflects how well the model separates the two classes. In general, a higher AUC(ROC) score generally results in better performance for a machine learning model. When AUC = 1, two classes can be distinguished perfectly (Figure 4.2).

Figure 4.2
AUC ROC Curve



A ROC AUC is plotted with the TP rate on the y axis and the FP rate on the x axis (Figure 4.3).

Figure 4.3
True Positive Rate vs False Positive Rate



4.3. Model Result: Original Dataset

Four classifiers were used for classification, running the algorithm on the original dataset from Kaggle. Table 4.1, Figure 4.4, and Figure 4.5 show the results of the four classifiers. The accuracy of the majority of the classifiers was 99.98%. However, the appropriate model should be selected based on AUC performance. In terms of AUC scores, random forest performed best, with the highest score of the classifiers. However, these results cannot be trusted because the model was trained on a highly imbalanced dataset. Thus, bias was introduced into the results, resulting in poor performance. Data preparation steps such as oversampling and undersampling should be performed before any machine learning program is trained if the target distribution is not evenly distributed.

Table 4.1
Scores for the Original Dataset

Model	Accuracy	AUC	Precision score	Recall score	F1 score
Logistic regression	0.998104	0.5	0	0	0
Decision tree	0.999204	0.873298	0.817568	0.746914	0.780645
Random forest	0.999544	0.885791	0.984252	0.771605	0.865052
Autoencoders with DNN	0.998361	0.651076	0.644737	0.302469	0.411765

Figure 4.4
AUC Score for the Original Dataset

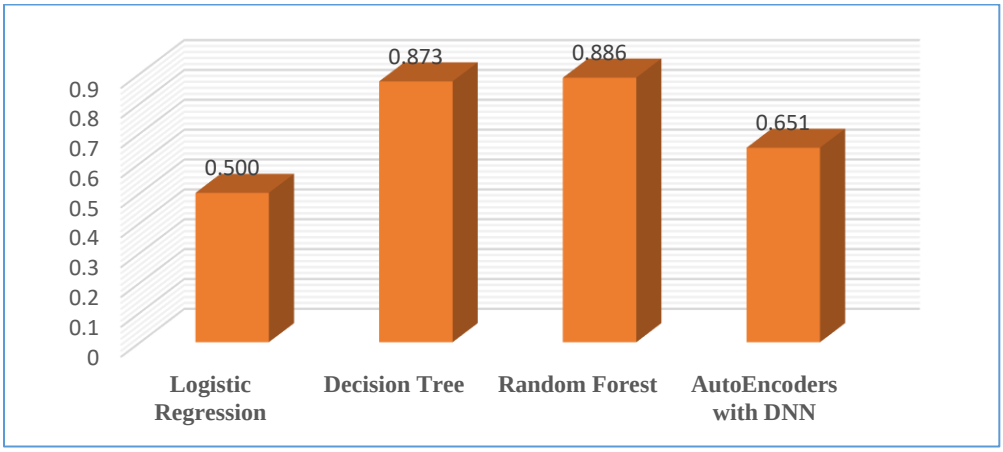
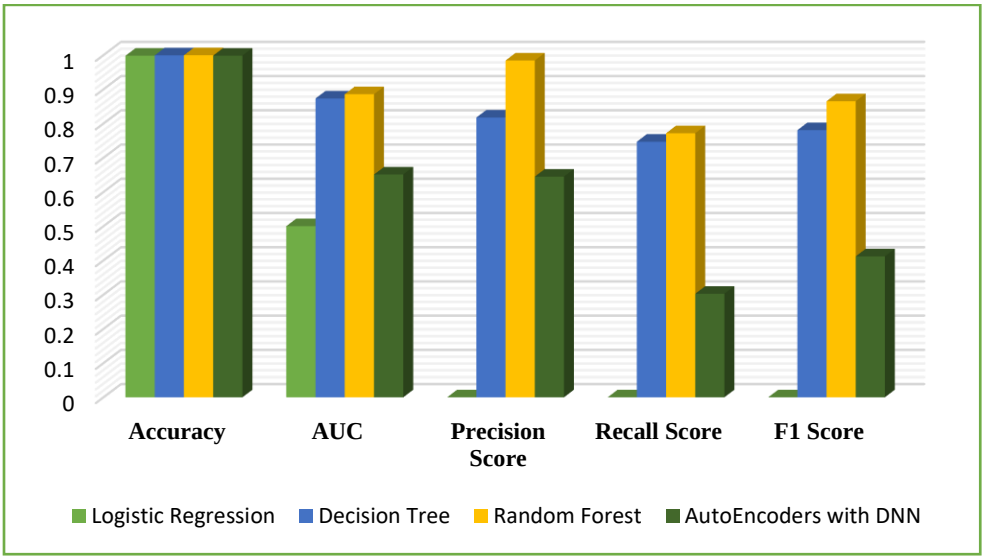


Figure 4.5
Metrics for the Original Dataset



4.4. Model Result: Undersampling Dataset

Data resampling was used to resolve the imbalance in the dataset. To match the minority class with the majority class, random undersampling was performed to account for the difference between the two classes. With the undersampled data, four classifiers were trained and then tested. Results are shown in Table 4.2, Figure 4.6, and Figure 4.7.

With the undersampled dataset, the accuracy for all the classifiers was the same as the accuracy obtained with the original dataset, with the exception of the decision tree, which had a lower accuracy. When there is imbalance in the data, the AUC score is the most important performance metric for the classifier. The random forest was able to perform better than any other algorithm; it achieved a score of 93.20 out of 100. When using logistic regression, a supervised classification method, undersampling generated the worst results.

Table 4.2
Scores for the Undersampling Data Model

Model	Accuracy	AUC	Precision score	Recall score	F1 score
Logistic regression	0.998104	0.5	0	0	0
Decision tree	0.887059	0.897214	0.015028	0.907407	0.029566
Random forest	0.950446	0.932048	0.033883	0.91358	0.065342
Autoencoders with DNN	0.997179	0.888873	0.211679	0.179012	0.19398

Figure 4.6
AUC Scores for the Undersampling Data Model

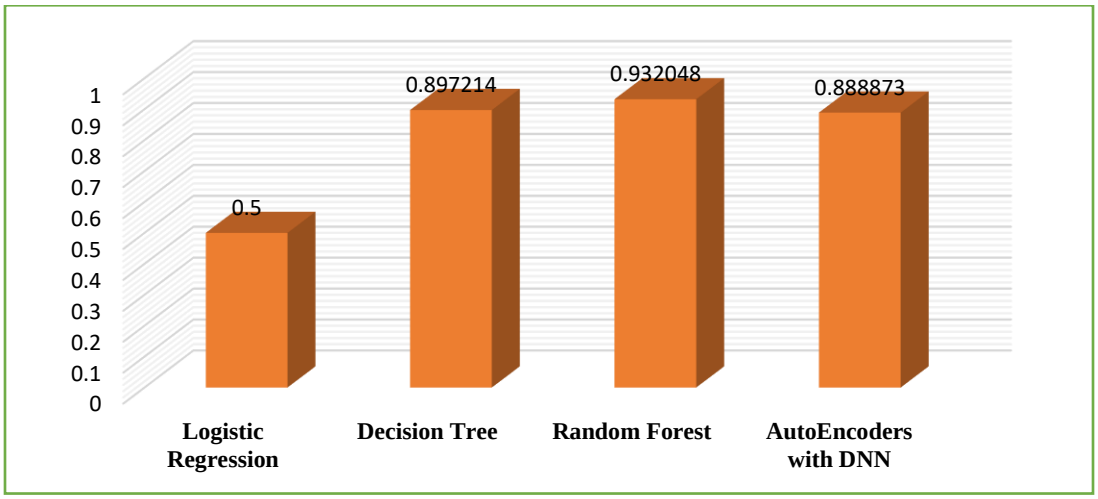
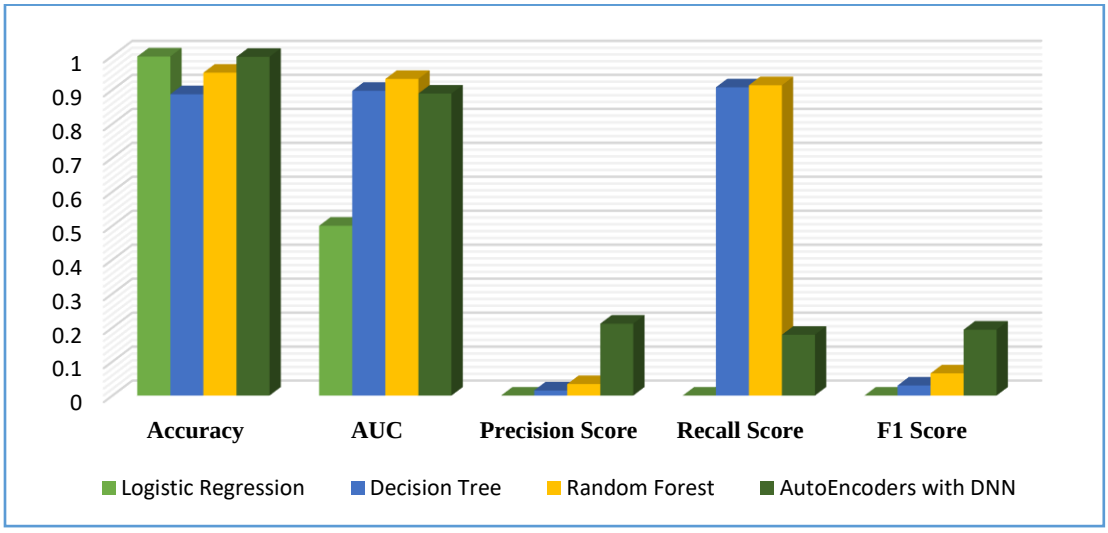


Figure 4.7
Metrics for the Undersampling Data Model



4.5. Model Result: Oversampling Dataset (SMOTE)

SMOTE is an oversampling technique used to generate samples by finding a K-nearest neighbor cluster in the feature space and selecting a point within the cluster as the

point at which the sample is to be created. The value of K equal to 5 was selected for this case. After oversampling the data using SMOTE, the data were used to train using four classifiers: random forest, logistic regression, decision tree, and autoencoder with DNN. Table 4.3, Figure 4.8, and Figure 4.9 show the results. Clearly, the autoencoder with DNN had a better AUC and better accuracy than the other classifiers. In addition, SMOTE oversampling produced better results than other sampling techniques. Overall, combining SMOTE with an autoencoder to detect fraud in credit card transactions gave better results than other techniques.

Table 4.3
Scores for the Oversampling (SMOTE) Data Model

Model	Accuracy	AUC	Precision score	Recall score	F1 score
Logistic regression	0.996278	0.788658	0.273256	0.580247	0.371542
Decision tree	0.997952	0.894235	0.475836	0.790123	0.593968
Random forest	0.999508	0.907337	0.916667	0.814815	0.862745
Autoencoders with DNN	0.997540	0.994387	0.996954	0.998147	0.99755

Figure 4.8
AUC Scores for the Oversampling (SMOTE) Data Model

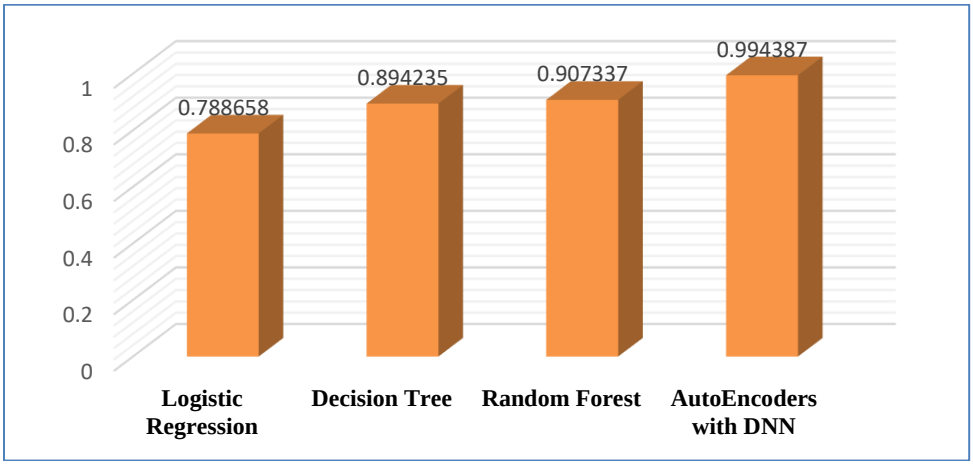
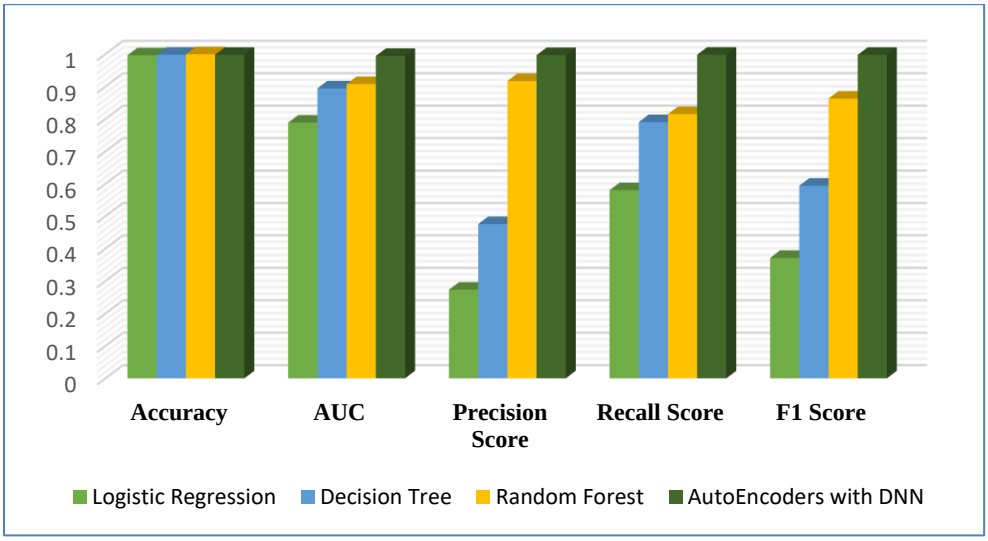


Figure 4.9
Metrics of the Oversampling (SMOTE) Data Model



4.6. Model Result: Oversampling Dataset (ADASYN)

ADASYN is very similar to SMOTE in that it generates synthetic samples for minorities, but it uses weight distributions for minority observation types that are more difficult to learn. Also, it improves learning by reducing bias introduced by class imbalance.

Based on the results of model performance in Table 4.4, Figure 4.10, and Figure 4.11, it can be concluded that all algorithms performed very well with oversampling datasets compared to undersampling datasets. Adaptive oversampling techniques also performed better than undersampling techniques and the original dataset, but the SMOTE sampling method outperformed ADASYN based on the AUC score for the same classifiers.

Table 4.4
Scores for the Oversampling (ADASYN) Data Model

Model	Accuracy	AUC	Precision score	Recall score	F1 score
Logistic regression	0.996161	0.816324	0.276882	0.635802	0.385768
Decision tree	0.997893	0.884964	0.466418	0.771605	0.581395
Random forest	0.999508	0.907337	0.916667	0.814815	0.862745
Autoencoders with DNN	0.981883	0.911635	0.087008	0.901235	0.158696

Figure 4.10
AUC Scores for the Oversampling (ADASYN) Data Model

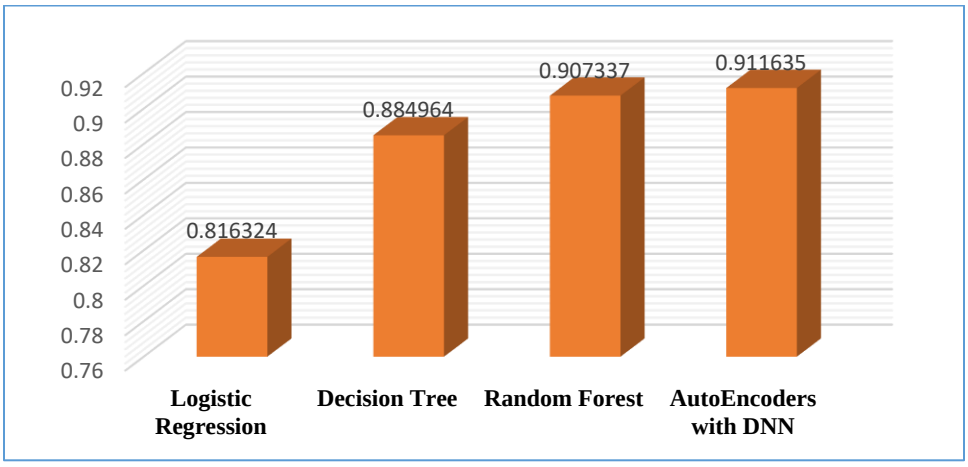
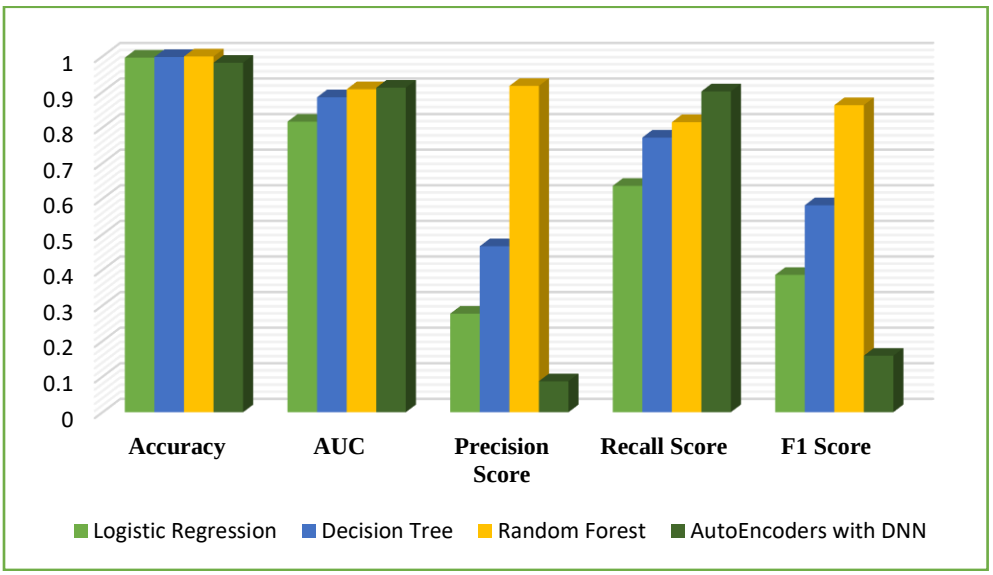


Figure 4.11
Metrics of the Oversampling (ADASYN) Data Model



4.7. Comparative Analysis

This study evaluated four different classification methods—random forest, logistic regression, decision tree, and autoencoder with DNN—on oversampled and undersampled datasets. The model was tested with a different dataset, and the AUC score was used to evaluate performance and identify the most accurate overall model. In most of the cases, oversampling with SMOTE was used in conjunction with autoencoders with DNNs and evaluated on different parameters like AUC score, recall, precision, and F1 score. Figures 4.12 to 4.15 represent the confusion matrices on the testing data

Figure 4.12
Model Name: Logistic Regression SMOTE

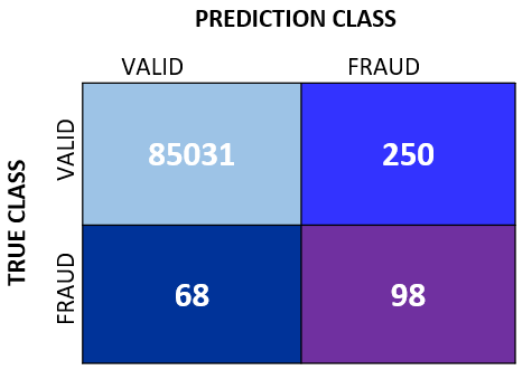


Figure 4.13
Model Name : Decision Tree SMOTE

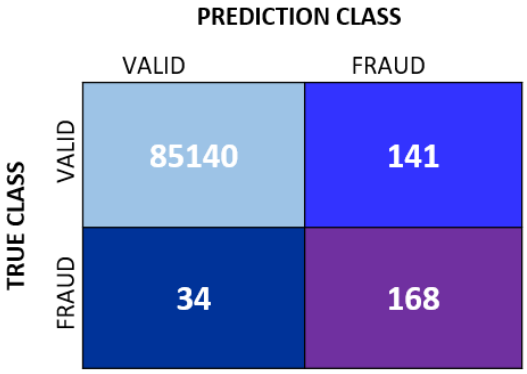


Figure 4.14
Model Name : Random Forest SMOTE

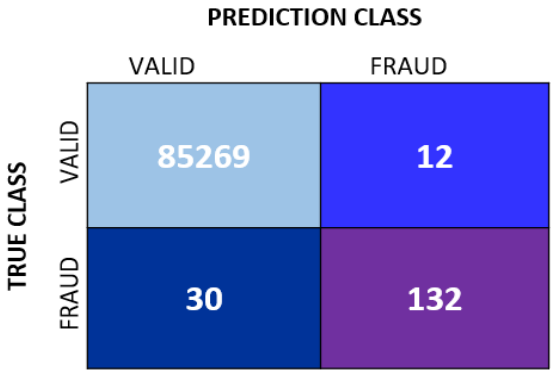
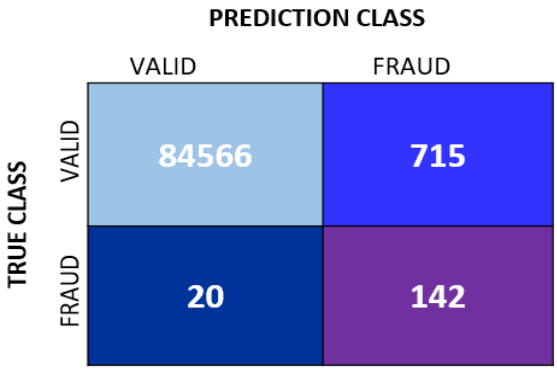


Figure 4.15
Model Name : Autoencoder with DNN SMOTE

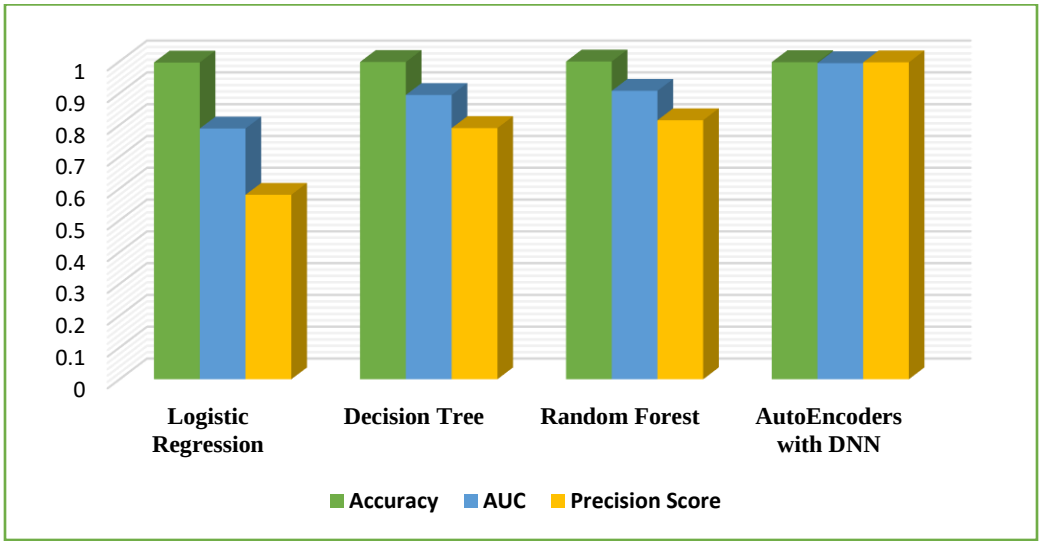


In addition, autoencoder with DNN had the highest AUC score, accuracy, and precision (Table 4.5, Figure 4.16). Hence, autoencoder with DNN was selected as the best performing predictive model to detect fraud in the credit card transactions based on AUC score. But Random forest performance is also very close to the hybrid approach. Depending on the need and requirement of the company one can make the decision between hybrid approach random forest (on the basis of FP and FN – Cost of FN and Cost of FP).

Table 4.5
Comparison of Scores for the Different Models

Model	Accuracy	AUC	Precision score
Logistic regression	0.996278	0.788658	0.580247
Decision tree	0.997952	0.894235	0.790123
Random forest	0.999508	0.907337	0.814815
Autoencoders with DNN	0.99754	0.994387	0.996954

Figure 4.16
Comparison of Metrics for the Different Models



By applying AUC metrics to all the results, the conclusion is that autoencoders with DNNs performed better than other classifiers based on AUC metrics. To predict credit card fraud, autoencoder with DNN was the most effective method on AUC Score.

4.8. Hypothesis and Results

Below is a description of the three hypotheses and their validation.

H1: The SMOTE technique for class imbalance provides statistically better results than Undersampling and ADASYN.

After analyzing undersampling and oversampling techniques for credit class imbalances, SMOTE proved to be the best oversampling technique for credit class imbalances. Due to the nature of the algorithm, it provides a major advantage, as it prevents the problem of overfitting that is caused by random oversampling. Interpolating between positive instances in the feature space that are close to one another creates new instances by generating new regions in the feature space as a result of the interpolation.

H2: Ensemble technique–based classifiers like random forest give better AUC results than classification models like logistic regression and decision tree.

Ensemble techniques like random forest performed better than logistic regression and decision tree. An ensemble of predictive models performs better when it comes to predicting future events than a single predictive model that is used alone. The variance component of error can be reduced by adding bias to the model before making a prediction to reduce the variance component of the prediction error.

H3: A hybrid method that combines supervised and unsupervised techniques improves fraud detection accuracy.

A hybrid model with a combination of supervised and unsupervised techniques outperformed the other algorithms. Autoencoder with neural network performed better than other machine learning algorithms because autoencoder is able to learn nonlinear transformations through a nonlinear activation function and multiple layers.

A deep learning-based autoencoder can be used to replace existing models as a way to overcome the inefficiency of those models. As far as determining whether a given transaction is genuine or fraudulent, there are several methods available, including logistic regression and decision trees. However, the results showed that the proposed model outperformed these other methods. As companies are collecting large amounts of data, deep learning techniques such as neural networks and autoencoders are a smart idea because they can be combined to produce better hybrid predictive models for fraud detection. Deep learning can also assist in finding different patterns that can help improve the accuracy of predictions in a manner that is even more efficient.

CHAPTER 5:

DISCUSSION AND CONCLUSION

5.1. Discussion

This praxis investigated whether the unsupervised learning method autoencoder combined with the supervised learning method deep neural network (DNN) was effective in solving the problem of credit card fraud. Four models were developed for the highly imbalanced dataset. In the proposed model, autoencoding was used, and the idea was very straightforward. By using the synthetic minority oversampling technique (SMOTE), a dataset that was imbalanced was transformed into a dataset that was balanced. The autoencoder was then used to remove noise from the sampled dataset. As a final step, fully connected DNNs were used to train the model, and that model was used to classify whether the given transaction was fraudulent or not. A model without oversampling or autoencoding had a very low recall rate and low area under the curve (AUC), because it classified all sample data as normal, so most fraud transactions went undetected. Most fraud transactions can be detected with an acceptable recall rate along with high AUC score using an oversampled and autoencoder model.

As a result of fraud detection mechanisms being present, it is easier for merchants and cardholders to build trust with each other. A more effective fraud detection system can decrease the amount of funds that need to be invested for manual fraud detection. The detection system will reduce the loss of jobs. Great care must be taken when implementing a system that detects fraud, since many factors can affect its effectiveness.

5.2. Conclusions

Payment networks are systems, requiring a scientifically eclectic engineering approach to design fault tolerant payment systems. Because exogenous actors—fraudsters—have demonstrated increasing sophistication in their fraud attempts, financial institutions must push the envelope, via systems science principles, to counter these efforts at propagating errors or faults (fraud) into the payment processing system, adding complexity to the overall system.

The work presented in this praxis demonstrates how a focus on the engineered estimation algorithms alone is not sufficient to build robust payment processing systems. Improving prediction accuracy also requires attention to structuring, conforming, and curation of the source—the training and testing data used in building and maintaining these payment systems. Hence, this work contributes to the body of knowledge in systems engineering sciences to identify and formalize approaches to surface anticipated errors that these engineered systems will encounter.

Specifically, autoencoder and deep neural networks—estimation algorithms—can be used to more effectively detect fraud in credit card transactions in large volumes of transactions. This achievement of improved accuracy in detecting fraud will increase trust between companies and customers. Nevertheless, machine learning and deep learning algorithms still face the data challenge of class imbalance. Resampling of original records is one approach that can be used to overcome this problem in preparing testing and training data sets for estimation, across machine learning and DNN algorithms. The study findings have demonstrated that the oversampling technique SMOTE is an effective way of solving the class imbalance problem and, when combined with autoencoder and DNN

algorithms, generates superior fraud detection capabilities for use in credit card payment systems.

5.3. Recommendations for Future Research

A few things can be done to increase the effectiveness of fraud detection in credit card transactions. Autoencoders come in many types and versions—denoise autoencoders, sparse autoencoders, and so on—all of which can be used in the coding process. Apart from autoencoders, restricted Boltzmann techniques can also be used. Restricted Boltzmann machines are generative stochastic neural networks capable of learning probability distributions over their inputs.

SVM SMOTE and BorderLine SMOTE are two oversampling methods that can be used to resample the data prior to training the classifier. It may be possible to achieve better results by using these techniques. The use of interpolation is also an alternative to oversampling methods. Different combinations of layers and activation functions can be used to improve the precision and accuracy of the prediction model.

REFERENCES

Abreu, R., David, F., & Segura, L. (2016). E-banking services: Why fraud is important? *Journal of Information Systems Engineering and Management, 1*(2), 111-121. https://doi.org/10.20897/lectito.201617

Adepoju, O., Wosowei, J., & Jaiman, H. (2019). Comparative evaluation of credit card fraud detection using machine learning techniques. *2019 Global Conference for Advancement in Technology (GCAT).* IEEE. https://doi.org/10.1109/GCAT47503.2019.8978372

Alenzi, H. Z., & Aljehane, N. O. (2020). Fraud detection in credit cards using logistic regression. *International Journal of Advanced Computer Science and Applications, 11*(12), 540-551. https://thesai.org/Downloads/Volume11No12/Paper_65-Fraud_Detection_in_Credit_Cards.pdf

Al-Rahman Al-Serw, N. (2021, February 21). Undersampling and oversampling: An old and a new approach. *Analytics Vidhya.* https://medium.com/analytics-vidhya/undersampling-and-oversampling-an-old-and-a-new-approach-4f984a0e8392

Al-Smadi, B. (n.d.). Credit card security system and fraud detection algorithm. Louisiana Tech Digital Commons. Retrieved July 26, 2022, from https://digitalcommons.latech.edu/dissertations/924/

Awoyemi, J., Adetunmbi, O., & Oluwadare, S. (2017). Credit card fraud detection using machine learning techniques: A comparative analysis. *2017 International Conference on Computing Networking and Informatics (ICCNI).* IEEE. https://doi.org/10.1109/ICCNI.2017.8123782

Bagga, S., Goyal, A., Gupta, N., & Goyal, A. (2020, July 1). Credit card fraud detection using pipeling and ensemble learning. *Procedia Computer Science, 173,* 104-112. https://doi.org/10.1016/j.procs.2020.06.014

Bandyopadhyay, H. (2022). Autoencoders in deep learning: Tutorial & use cases. *V7Labs*. https://www.v7labs.com/blog/autoencoders-guide

Bhattacharyya, S., Jha, S., Tharakunnel, K., & Westland, J. (2011). Data mining for credit card fraud: A comparative study. *Decision Support Systems, 50*(3), 602–613. https://doi.org/10.1016/j.dss.2010.08.008

Brownlee, J. (2020). How to calculate precision, recall, and f-measure for imbalanced classification. *Machine Learning Mastery.* https://machinelearningmastery.com/precision-recall-and-f-measure-for-imbalanced-classification/

Ghosh, S., & Reilly, D. L. (1994). Credit card fraud detection with a neural-network. *1994 Proceedings of the Twenty-Seventh Hawaii International Conference on System Sciences, 3,* 621-630. https://www.semanticscholar.org/paper/Credit-card-fraud-detection-with-a-neural-network-Ghosh-Reilly/ba70a74262adec9dcfa47b5710752d2537a07af4

Gupta, S., & Johari, R. (2011). A new framework for credit card transactions involving mutual authentication between cardholder and merchant. *2011 International Conference on Communication Systems and Network Technologies* (pp. 22-26). https://doi.org/10.1109/CSNT.2011.12

Hatami, M., Zahraee, S. M., Ahmadi, M., Golroudbary, S. R., & Rohani, J. M. (2014). Improving productivity in a bank system by using computer simulation. *Applied*

Mechanics and Materials, 606, 259–263. https://doi.org/10.4028/www.scientific.net/amm.606.259

IBM Cloud Education. (2020a, December 7). Random forest. https://www.ibm.com/cloud/learn/random-forest

IBM Cloud Education. (2020b). What is deep learning? IBM. https://www.ibm.com/cloud/learn/deep-learning

Jaiswal, S. (n.d.). Decision tree classification algorithm. JavaTPoint. Accessed July 20, 2022. https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm

James, J. (2003). Bayes' theorem. *Stanford Encyclopedia of Philosophy*. https://plato.stanford.edu/entries/bayes-theorem/

Jiang, C., Song, J., Liu, G., Zheng, L., & Luan, W. (2018, October). Credit card fraud detection: A novel approach using aggregation strategy and feedback mechanism. *IEEE Internet of Things Journal, 5*(5), 3637–3647. https://doi.org/10.1109/JIOT.2018.2816007

John, G., & Langley, P. (2013). *Estimating continuous distributions in Bayesian classifiers*. arXiv preprint1302.4964.

Kavlakoglu, E. (2020). AI vs. machine learning vs. deep learning vs. neural networks: What's the difference? *IBM.* https://www.ibm.com/cloud/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks

Kiernan, J. S. (2022, May 12). Credit card and debit card fraud statistics. *Wallethub*. https://wallethub.com/edu/cc/credit-debit-card-fraud-statistics/25725/

Krishna, M. (2021). Equifax hack: 5 biggest credit card data breaches. *Investopedia*. https://www.investopedia.com/news/5-biggest-credit-card-data-hacks-history/

Kumar, M. S., Soundarya, V., Kavitha, S., Keerthika, E. S., & Aswini, E. (2019). Credit card fraud detection using random forest algorithm. *Proceedings of the 3rd International Conference on Computing and Communications Technologies (ICCCT), Chennai, India* (pp. 149–153). https://doi.org/10.1109/ICCCT2.2019.8824930

Lakshmi, S., & Kavilla, S. (2018). Machine learning for credit card fraud detection system. *International Journal of Applied Engineering Research, 13*(24), 16819–16824.

Lucas, Y. (2019). *Credit card fraud detection using machine learning with integration of contextual knowledge.* Artificial Intelligence [cs.AI], Université de Lyon, Universität Passau (Deutscheland).

Lucas, Y., Portier, P.-E., Laporte, L., He-Guelton, L., Caelen, O., Granitzer, M., & Calabretto, S. (2020). Towards automated feature engineering for credit card fraud detection using multi-perspective HMMs. *Future Generation Computer Systems, 102*, 393-402. https://doi.org/10.1016/j.future.2019.08.029

Luo, G., Li, W., & Peng, Y. (2020). Overview of intelligent online banking system based on HERCULES architecture. *IEEE Access, 8,* 107685-107699. https://doi.org/10.1109/ACCESS.2020.2997079.

Maes, S., Tuyls, K., Vanschoenwinkel, B., & Manderick, B. (2002). Credit card fraud detection using Bayesian and neural networks. *Proceedings of the 1st International Naiso Congress on Neuro Fuzzy Technologies* (pp. 261-270).

Makki, S., Assaghir, Z., Taher, Y., Haque, R., Hacid, M., & Zeineddine, H. (2019). An experimental study with imbalanced classification approaches for credit card fraud detection. *IEEE Access: Practical Innovations, Open Solutions, 7*, 93010–93022. https://doi.org/10.1109/ACCESS.2019.2927266

Ng, A., & Jordan, M. (2002). On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. *Advances in Neural Information Processing Systems, 2*, 841–848.

Nilson Report. (2019, November 18). Card fraud losses reach $27.95 billion. https://nilsonreport.com/mention/407/1link/

Pozzolo, A. D. (2015). *Adaptive machine learning for credit card fraud detection.* Dissertation, University of Brussels. https://dalpozz.github.io/static/pdf/Dalpozzolo2015PhD.pdf

ProjectPro. (2022). Deep learning vs machine learning—what's the difference? https://www.projectpro.io/article/deep-learning-vs-machine-learning-whats-the-difference/414

Prusti, D., & Rath, S. K. (2019). Web service-based credit card fraud detection by applying machine learning techniques. *Proceedings of the TENCON 2019—2019 IEEE Region 10 Conference (TENCON), Kochi, India* (pp. 492–497). https://doi.org/10.1109/TEN- CON.2019.8929372

Quinlan, R. (1986). Induction of decision trees. *Machine Learning, 1*(1), 1–106.

Rout, A. R. (2020). Advantages and disadvantages of logistic regression. *Geeks for Geeks.* https://www.geeksforgeeks.org/advantages-and-disadvantages-of-logistic-regression/

Sadgali, I., Sael, N., & Benabbou, F. (2019). Fraud detection in credit card transaction using neural networks. *Proceedings of the 4th International Conference on Smart City Applications (SCA '19)* (pp. 1-4). Association for Computing Machinery. https://doi.org/10.1145/3368756.3369082.

Sahin, Y., & Duman, E. (2011a). Detecting credit card fraud by ANN and logistic regression. *2011 International Symposium on Innovations in Intelligent Systems and Applications* (pp. 315-319). IEEE. https://doi.org/10.1109/INISTA.2011.5946108

Sahin, Y., & Duman, E. (2011b). Detecting credit card fraud by decision trees and support vector machines. *Proceedings of International Multi-Conference of Engineers and Computer Scientists (IMECS 2011)* (vol. 1, pp. 1-6).

Saini, A. (2021). Support vector machine (SVM): A complete guide for beginners. *Analytics Vidhya*. https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/

Sarantitis, G. (2020, May 14). Fraud detection using autoencoders. https://gsarantitis.wordpress.com/2020/05/14/fraud-detection-using-autoencoders/

Save, P., Tiwarekar, P., Jain, K., & Mahyavanshi, N. (2017). A novel idea for credit card fraud detection using a decision tree. *International Journal of Computer Applications, 161*(13), 6–9. https://doi.org/10.5120/ijca2017913413

Seera, M., Lim, C. P., Kumar, A., Dhamotharan, L., & Tan, K. H. (2021). An intelligent payment card fraud detection system. *Annals of Operations Research,* online before print. https://doi.org/10.1007/s10479-021-04149-2

Sharma, A. (2018, November 14). Differences between machine learning & deep learning. *DataCamp*. https://www.datacamp.com/tutorial/machine-deep-learning

Sherly, K. (2012). A comparative assessment of supervised data mining techniques for fraud prevention. *TIST International Journal of Science, Technology and Research, 1*(1), 1-6.

Shift Credit Card Processing. (2021). Credit card fraud statistics. Updated January 2021. https://shiftprocessing.com/credit-card-fraud-statistics/

Silver, C. (2019, July 29). Capital One data breach impacts 106 million customers. *Investopedia.* https://www.investopedia.com/capital-one-reveals-massive-hack-exposingmillions-4707235

Sohony, I., Pratap, R., & Nambiar, U. (2018). Ensemble learning for credit card fraud detection. *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data Association for Computing Machinery* (pp. 289–294). https://doi.org/10.1145/3152494.3156815

Suryanarayana, S. V., Balaji, G. N., & Venkateswara Rao, G. (2018). Machine learning approaches for credit card fraud detection. *International Journal of Engineering & Technology, 7*(2), 917-920. http://dx.doi.org/10.14419/ijet.v7i2.9356

TIBCO. (n.d.). What is a random forest? Retrieved July 24, 2022, from https://www.tibco.com/reference-center/what-is-a-random-forest.

Trivedi, N. K., Simaiya, S., Lilhore, U. K., & Sharma, S. K. (2020). An efficient credit card fraud detection model based on machine learning methods. *International Journal of Advanced Science and Technology, 29*(5), 3414–3424.

Wikimedia Foundation. (2022, June 26). Systems engineering. Wikipedia. Retrieved July 26, 2022, from https://en.wikipedia.org/wiki/Systems_engineering

Xuan, S., Liu, G., Li, Z., Zheng, L., Wang, S., & Jiang, C. (2018). Random forest for credit card fraud detection. *2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC)* (pp. 1-6). https://doi.org/10.1109/ICNSC.2018.8361343

Yeşilkanat, A., Bayram, B., Köroğlu, B., Arslan, S. (2020). An adaptive approach on credit card fraud detection using transaction aggregation and word embeddings. In I. Maglogiannis, L. Iliadis, & E. Pimenidis (Eds.), *AIAI 2020: Artificial intelligence applications and innovations*, 3-14. Springer. https://doi.org/10.1007/978-3-030-49161-1_1

APPENDIX: DATA AVAILABILITY

The raw data required to reproduce the above findings are available to download from https://github.com/Kumarsanjayanalytics/Kumarsanjayanalytics.