**CREDIT CARD FRAUD DETECTION USING DEEP LEARNING**

**BY**

**ABAYOMI-ALLI AYOMIDE**

**MATRIC NO: 20162994**

**DEPARTMENT OF COMPUTER SCIENCE**

**COLLEGE OF PHYSICAL SCIENCES**

**FEDERAL UNIVERSITY OF AGRICULTURE, ABEOKUTA (FUNAAB)**

**IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF BACHELOR OF SCIENCE DEGREE IN COMPUTER SCIENCE**

**SEPTEMBER, 2021**

**CERTIFICATION**

I hereby certify that the work recorded in this Project was carried out by **ABAYOMI-ALLI AYOMIDE TEMIDAYO** with **Matriculation Number 20162994** on **Credit Card Fraud Detection Using Deep Learning**. In partial fulfillment of the requirement of the award of Bachelor Degree in the Department of **COMPUTER SCIENCE** of the Federal University of Agriculture, Abeokuta, (FUNAAB), Ogun State, Nigeria.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ABAYOMI-ALLI AYOMIDE Date**

Student

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**MR. CHARLES UGWUNNA Date**

SupervisingLecturer

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**PROF. FOLORUNSO OLUWASEGUN Date**

Head of Department (H.O.D)

**DEDICATION**

This project work is dedicated to Almighty God for His protection over my life from the beginning of this journey up to this points.

I also dedicate this report to my dad Mr Abayomi-Alli and in loving memory of my mother, Late Mrs. Abayomi-Alli. And also, as well as my siblings and Dr. Abayomi-Alli Adebayo for their support, their financial assistance and encouragement all through this journey.

**ACKNOWLEDGEMENT**

I will like to give glory and honor to Almighty God for being my pillar, for being my strength through it all.

Also, I will like to thank and appreciate my project supervisor, Mr. Charles Ugwunna for being supportive and for his advice through-out the project work.

Special thanks to Dr. Abayomi-Alli Adebayo, Dr. (Mrs) O.E. Ojo and all the lecturers in the Department of Computer Science for their immerse support and dedication towards impacting knowledge in me.

Abstract

Today, fraudulent card payments continue to be a problem for e-commerce businesses and online banking operations, resulting in billions of dollars in annual losses. The development of an efficient algorithm is one of the most difficult challenges in this subject. The purpose of this research is to offer an effective method for automatically detecting credit card fraud involving financial institutions by utilizing a deep learning algorithm model called LSTM and an auto-encoder. Credit card transaction data was gathered from Kaggle, a data repository, and includes a total of 284,808 credit card transactions from a European bank data collection. It classifies fraudulent transactions as the "positive class" and non-fraudulent transactions as the "negative class." The data set is significantly skewed; around 0.172 percent of transactions are fraudulent, while the remainder are genuine. Two deep learning algorithms are used to analyze the dataset, and the work is carried out in Python using the Jupyter notebook IDE. The methods' performance is assessed for a variety of factors, including accuracy, execution time, and loss rate. The result indicates that the LSTM and Auto-Encoders have an accuracy of 99.98 and 70.24, respectively. The comparison findings indicate that the LSTM approach outperforms the Auto-Encoder technique.

**TABLE OF CONTENT**

Cover Page i

Certification Page ii

Dedication iii

Acknowledgement iv

Abstract v

Table of Content vi

**CHAPTER ONE**

**INTRODUCTION**

1.1 Background of the Study 1

1.2 Statement of the Problem 2

1.3 Aims and Objectives 3

1.4 Significance of the Study 3

1.5 Scope of Study 4

1.6 Limitation of Study 5

1.7 Basic Terms 5

**CHAPTER TWO**

**LITERATURE REVIEW**

2.1 Conceptual Framework 8

2.2 Review of Related Works 13

**CHAPTER THREE**

**RESEARCH METHODOLOGY**

3.1 Design Consideration 18

3.2 Architectural Framework 20

3.3 Auto-encoder 23

3.4 Evaluation Metrics 24

3.5 Psuedocodes for LSTM and Auto-Encoder 26

**CHAPTER FOUR**

**IMPLEMENTATION AND EVALUATION**

4.1 Hardware & Software Requirements 28

4.2 Choice of Programming Environment 28

4.3 System Testing 28

4.4 Limitation of the System 29

4.5 Results and Discussion 29

**CHAPTER FIVE**

**SUMMARY, CONCLUSION AND RECOMMENDATIONS**

5.1 Conclusion 33

5.2 Recommendation 33

5.3 Contribution to Knowledge 33

5.4 Future Work 33

**REFERENCES 34**

**APPENDIX 37**

**CHAPTER ONE**

1. **Background of Study**

Over the previous few decades, Internet usage has exploded, resulting in an explosion of E-commerce, online services, online advertising, and online shopping. Essentially, the majority of businesses have migrated to the Internet, and electronic payment transactions occur on a regular basis around the world, more frequently than alternative payment methods such as money transfer via online banking systems or crypto-currency. Individuals may now book appointments online, pay bills online, and enroll in paid programs online, which simplifies our lives significantly by eliminating the need to queue at the bank or carry cash around. Nonetheless, all of these thieves have infiltrated the e-commerce system using futuristic methods of credit card theft. On a daily basis, there is either a theft of credit card information, an illegal transaction, or the use of a counterfeit card. Globally, this increasing fraud has resulted in huge losses, necessitating the creation of mechanisms to identify it. While machine learning and data mining techniques are being utilized to identify credit card fraud, they are still in need of development due to their inability to produce significantly realistic results. However, statistically significant methods that are efficient and realistic should be created. In this example, a deep learning method is utilized to detect fraudulent transactions before they are approved by the appropriate authorities (Yomas & Kiran, 2018).

Credit card fraud detection is a more sophisticated form of theft; it occurs when an unauthorized individual steals information from a credit card or gains access to the credentials of another cardholder for illegal purposes, impersonation, and dubious activities. These fraudulent activities can be physical, in which the fraudster physically uses the credit card to make purchases, or virtual, in which the fraudster uses the credit card to make virtual purchases. Fraudsters commit such acts through the following methods: misplaced or stolen cards, card cloning and scanning, in which fraudsters copy information from the magnetic strip and create a card that looks identical to the original card, hijacked account, in which fraudsters take complete control of the user's account in order to commit fraud, stolen credit card id, card holder not present (Andrea et al., 2005). Additionally, fraudsters employ additional techniques on the Internet, such as site copying, misleading trade sites, and illicit credit card manufacturing (Odumuyiwa et al., 2019).

Global financial losses incurred as a result of credit card theft were 22.8 billion US dollars in 2017 and were expected to continue increasing (Lebichot et al., 2019). Additionally, the Nilson research claimed that gross losses from credit card theft are anticipated to exceed forty billion US dollars in 2027, up from 28.65 billion in 2019. (Nilson Report, 2021) To mitigate these losses, detection and preventive mechanisms should be implemented. Mechanisms for distinguishing fraudulent from genuine transactions have been created utilizing machine learning and data mining techniques. Machine learning (ML) is the study of algorithms that improve automatically as a result of training data, such that a computer can perform a task by testing the trained data. Machine learning is the most widely used technology because it is more accurate, consumes less time, and is applicable to a wide variety of applications. On the other hand, deep learning is a subset of machine learning that consists of neural networks that analyze data and make decisions in a manner similar to the human brain. Several deep learning models include the convolutional neural network (CNN), the recurrent neural network (RNN), the artificial neural network (ANN), the long short term memory (LSTM) network, stack auto-encoders, the deep boltzman machine, multilayered perceptions, and the radial basis function network, among others.

**1.2 Statement of the problem**

The illegal financial operations are very sophisticated and difficult to detect. Frauds are rising significantly as a result of recent technological advancements, most notably in the financial industry (Tripathi and Pavaskar, 2012). Fraud offenses cost financial institutions billions of dollars each year, eroding the institution's financial health and, therefore, consumer confidence (West and Bhattacharya, 2016). Worldwide, financial institutions and companies are suffering significant losses as a result of many financial scams.

Credit card fraud is on the rise, as it is a widely utilized payment method. This is due to the progress of technology and the growth of online payment systems, which has resulted in a massive loss globally. As a result, there is a need for effective methods of mitigating the loss. Additionally, fraudsters have devised several methods for illegal access to credit card user information, including spoofing SMS and phone calls, duplicating websites, impersonating attacks, and spoofing applications utilizing Google forms (Asha and Suresh, 2021). In Nigeria, there has been a dramatic surge in the unlawful solicitation of sensitive information such as passwords and Bank Verification Numbers (BVNs) via phone calls, SMS message reset or code verification, and phishing email communications. Due to the increased pace at which products and services are purchased digitally using credit cards, credit card fraud has grown. The development of a viable model capable of reliably detecting fraudulent monetary transactions is critical.

**1.3 Aim and Objectives of the Study**

The primary objective of this research is to employ deep learning algorithms to identify and detect fraudulent credit card transactions.

The aims of this research are as follows:

i. To construct a model utilizing the LSTM method.

ii. To conduct a comparative analysis of LSTM, Auto encoder, and other established machine learning algorithms in the identification of credit card fraud.

**1.4 Significance of the Study**

Fraudulent transactions detection has long been a priority for both cardholders and financial institutions. Due to the fact that detecting even a small number of dishonest transactions may protect large sums of money, credit card theft has become a big issue for academics. As a result, this research will concentrate on the popularity and use of technology, particularly in the area of fraud detection. The dominant study's primary objective was to address a difficulty for machine learning, namely that the distribution of data regularly changes over time owing to new attack techniques and seasonality, and that a vanishingly small proportion of all credit card transactions is fraudulent.

However, machine learning techniques should be upgraded in terms of computing value, memory consumption, and processing vast amounts of data. The detection of financial fraud is a difficult task for a variety of reasons, including the following: fraudulent behavior is constantly changing, there is no mechanism for fine-tuning the statistics of the fraudulent transaction, existing detection strategies (such as machine studying algorithms) have significant limitations, and financial fraud datasets are extremely skewed, making it extremely difficult to train algorithms. The use of deep learning techniques remains limited, and approaches such as CNN and LSTM are advocated for image classification and Natural Language Processing (NLP), respectively, due to their capacity to handle massive datasets (Thanh et al., 2020). Additionally, this work makes a contribution by proposing a fraud detection model for detecting fraudulent credit card transactions that is entirely based on deep learning techniques (Long Short – Term Memory).

**1.5 Scope of the Study**

Deep learning will be used to do research on credit card fraud detection. Deep learning is a sophisticated machine learning approach in artificial intelligence that use neural networks to learn unsupervised from unstructured or unlabeled data. While many machine learning algorithms have been used to credit card fraud detection in a variety of scenarios and have produced reliable results, one restriction is that the algorithms are not well-suited to processing big datasets. With an accuracy of 97-99 percent, Support Vector Machines (SVM), K-Nearest Neighbor (k-NN), and K-means are the most frequently used anomaly detection algorithms in Machine Learning. However, due to the enormous quantity of data to be processed, deep learning techniques will be employed and the Long Short Term Memory Network (LSTM) method will be used.

The Long Short-Term Memory (LSTM) network is a kind of Recurrent Neural Network (RNN). A typical neural network cannot retain past knowledge and must execute the learning job from scratch each time. In the simplest terms, an RNN is a neural network that includes memory. Due to the fading gradient problem, RNNs have a tendency to exhibit short term memory. Back propagation is the foundation of neural networks since it minimizes loss by changing the network's weights based on gradients. In RNNs, when the gradient returns to the network, it decreases, resulting in a very tiny weight update. The previous layers of the network are unaffected by this little update, and the RNN loses its capacity to recall early instances in extended sequences, converting it to a short-term memory network.

The accuracy, loss rate, and execution time of the LSTM-based credit card fraud detection model will be compared to those of an existing machine learning-based model, such as the Support Vector Machine (SVM). Similarly, with the method of deep learning, Autoencoder.

**1.6 Limitation of Study**

Processing large amounts of data in order to provide correct results for fraud detection situations is time consuming and difficult. Due to the large memory consumption of the computations in comparison to the dataset utilized for the analysis, a high-performance machine is required. Additionally, there was inconsistency in the power supply and network disruption.

**1.7 Definition of Terms**

1. **Credit Card:** A financial card issued by financial institutions that allows card holders to borrow funds to settle purchase. It is a card that allows user to make payments for goods based on limit issued on the card by the financial institution.
2. **Credit Card Fraud:** An unauthorized intrusion or identity theft to a credit card. It is a fraudulent access and usage of card without the card-holder awareness and authorization for the purpose of making purchase and settling cash advances.
3. **Deep Learning:** is a part of machine learning in artificial intelligence that involves networks that are capable of learning unsupervised from unstructured or unlabeled input. Additionally, it is referred to as deep neural learning or network.
4. **E-Commerce:** Electronic commerce is the use of internet or online services and platform to buy and sell goods and services and perform financial transactions. It is also known as Internet Commerce.
5. **Fraud Detection:** It is attempt and series of activities undergone to discover and fetch out abnormalities and authorized access and usage of a financial item. It is undergone to detect suspicion and abnormal financial transactions.
6. **Keras:** is a Python application programming interface (API) for high-level neural networks. This open-source neural network library was created to facilitate rapid experimentation with deep neural networks. It is compatible with CNTK, TensorFlow, and Theano. Keras is committed to modularity, usability, and extensibility. It defers low-level computations to another library called the Backend. In mid-2017, TensorFlow accepted Keras and incorporated it. The tf.keras module (tensorflow.keras) provides access to it. The Keras library, on the other hand, may continue to work autonomously.

Keras is a framework for TensorFlow that acts as a wrapper. Thus, you may create a model using Keras's easier-to-use interface and then switch to TensorFlow when you need to utilize a feature that Keras lacks or when you need to access particular TensorFlow capabilities. As a result, you may easily incorporate TensorFlow code into the Keras training process or model.

1. **Machine Learning:** It is special type of Artificial Intelligence (AI) that make application (software) to accurately carry out decisions about the result or outcome of a concept. The algorithms of Machine Learning relies heavily on historical data (previous data) to predict and forecast the outcome of new set of values.
2. **MatplotLib:** is a widely used Python package for data visualization. It enables us to create charts and graphs in order to better comprehend and convey data visually. In matplotlib, there are two primary types of plots (Line and Scatter).

a. A line plot is created using a one-dimensional array.

b. The scatterplot makes use of a multi-dimensional array.

1. **Numpy:** Numpy Library deals with array called Numpy Arrays, which can be used to manipulate and calculate various mathematical calculations. Numpy provides the foundation for data structure structures and operations for Scipy. Those arrays are called ndarrays and are very efficient. The Numpy module is mainly used for working with numerical data. It provides us with a powerful object known as an Array. With Arrays, we can perform mathematical operations on multiple values in the Arrays at the same time, and also perform operations between different Arrays, similar to matrix operations.
2. **Pandas:** Pandas provides data structures functionalities that are used to manipulate and analyze data. Pandas for Machine Learning is divided onto two concepts called; Series and DataFrame in Data Structure. A Series is a one dimensional-array where the rows and column can be labelled, while Dataframe is a multidimensional array where the rows and columns can also be labelled.
3. **Scikit-learn:** Scikit-learn is probably the most useful library for machine learning in Python. The sklearn library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction. It is a library in Python that provides many unsupervised and supervised learning algorithms. It's built upon some of the technology be familiar with, like NumPy, Pandas, and Matplotlib.
4. **TensorFlow:** This is an open-sourced end-to-end platform, a library for multiple machine learning and deep learning tasks, while Keras is a high-level neural network library that runs on top of TensorFlow. Both provide high-level APIs used for easily building and training models, but Keras is more user-friendly because it’s built-in Python.

Researchers turn to TensorFlow when working with large datasets and object detection and need excellent functionality and high performance. TensorFlow runs on Linux, MacOS, Windows, and Android. The framework was developed by Google Brain and currently used for Google’s research and production needs. In summary, TensorFlow for machine learning applications and Keras for deep neural networks.

**CHAPTER TWO**

**LITERATURE REVIEW**

**2.1 Conceptual Framework**

Due to the economic and social implications of identifying fraudulent credit card transactions, the number of related studies and researches has increased. This phase summarizes several significant investigations.

**2.1.1 Previous Approaches to Credit Card Fraud Detection**

Although machine learning (ML) approaches have been used to detect credit card fraud, no fraud detection system has yet to achieve exceptional performance due to the fact that the distribution of data changes over time due to new attack approaches and seasonality, as well as the fact that a very small percentage of all credit card transactions are fraudulent (Dal Pozzolo et al., 2017). Recent advancements in deep learning have been made to address challenging challenges in a variety of fields.

Machine learning is a subset of Artificial Intelligence (AI) that develops models to make judgments about the end or consequence of a notion. ML algorithms largely rely on historical data (prior data) to anticipate and predict the outcome of a new set of values.

There are two primary approaches for identifying fraudulent credit card transactions using machine learning algorithms: supervised and unsupervised learning. Credit card transactions from the past are assessed as genuine or fraudulent using supervised learning algorithms. Then, this approach begins training on the data in order to develop a model capable of categorizing fresh datasets. By contrast, unsupervised learning algorithms are entirely dependent on the direct categorization of credit card transactions using what are considered to be typical patterns. The system then categorizes transactions that do not conform to these patterns as fraudulent credit card transactions (Altyeb and Sharaf, 2020). Both supervised and unsupervised learning techniques were used to identify credit card fraud (S. Bhattacharyya et al., 2011; M. Carminati et al., 2011).

The most well-known algorithms for detecting credit card fraud employ supervised learning and use categorized transactions to train their classifiers. Credit card fraud is identified using categorized characteristics derived from credit card transactions. Numerous machine learning techniques were used to identify fraudulent credit card transactions (R. Bolton and D. Hand, 2002). Numerous classification methods have been developed for the purpose of detecting fraudulent credit card transactions, including the following (P. Ravisankar et al., 2011):

Decision Tree: The Decision Tree algorithm is a member of the supervised learning algorithm family. In addition to addressing regression and classification issues, the decision tree approach, unlike other supervised learning algorithms, may be used to solve regression and classification problems.

The purpose of a Decision Tree is to develop a training model capable of predicting the class or value of a target variable by inferring basic decision rules from past data (training data).

Autoencoder: An autoencoder is a type of neural network that learns to replicate the input to the output. It contains an internal (hidden) layer that defines the code used to represent the input and is composed of two major components: an encoder that converts the input to the code and a decoder that converts the code to a reconstruction of the input. Perfectly doing the copying operation would just replicate the signal, which is why autoencoders are typically constrained in ways that compel them to estimate the input, retaining only the most significant features of the data in the copy.

Hidden Markov Models (HMMs): HMMs are a type of statistical model in which the studied system is considered to be a Markov process with hidden states. Dynamic programming methods may be used to estimate both the output emission probabilities from the hidden states and the transition probabilities between the hidden states using observable output sequences created by the Markov process.

K-Means Clustering: This is a method for Unsupervised Learning that divides the unlabeled dataset into distinct clusters. Here, K denotes the number of pre-defined clusters that must be generated during the procedure; for example, if K=2, two clusters will be created; if K=3, three clusters will be created; and so on. It is an iterative method that separates the unlabeled dataset into k distinct clusters, each of which contains just one dataset with identical features. However, it is a handy method for automatically discovering group categories in an unlabeled dataset without the requirement for training.

K-nearest neighbor (KNN) algorithm: The KNN algorithm is a straightforward supervised machine learning technique that may be used to tackle classification and regression issues. It is simple to construct and comprehend, but has the huge disadvantage of being much slower as the amount of data in use increases.

Logistic Regression (LR): In statistics, the logistic model (or logic model) is used to estimate the likelihood of occurrence of a certain class or event, such as pass/fail, win/lose, alive/dead, or healthy/sick. This may be expanded to simulate a variety of different types of events, for example, detecting whether a picture contains a cat, dog, or lion. Each identified object in the image would be assigned a probability between 0 and 1, with a cumulative probability of one.

Logistic regression is a statistical model that, in its simplest form, employs a logistic function to model a binary dependent variable. Mathematically, a binary logistic model has a dependent variable with two possible values, such as pass/fail, which is represented by an indicator variable with the values "0" and "1".

Nave Bayes (NB): The Nave Bayes algorithm is a supervised learning method based on the Bayes theorem that is used to solve classification issues. The Nave Bayes Classifier is a simple and effective Classification method that aids in the development of rapid machine learning models capable of making accurate predictions.

Random Forest (RF): Random forest uses almost identical hyperparameters as decision trees. While growing the trees, random forest adds extra unpredictability to the model. Rather of looking for the most critical feature when splitting a node, it looks for the best feature from a random selection of features.

Support Vector Machine (SVM): The Support Vector Machine (SVM) is a supervised machine learning method that may be used to solve both classification and regression problems. It is, however, mostly employed to solve categorization issues. SVM is a supervised machine learning model that makes use of classification techniques to solve issues involving two groups. After training using sets of labeled training data for each category, an SVM model is able to classify fresh data.

**2.1.2 Deep Learning Approach in Credit Card Fraud Detection**

Deep learning is a type of machine learning in artificial intelligence that use neural networks to learn unsupervised from unstructured or unlabeled data. Additionally referred to as deep neural network or deep neural learning. It emulates the human brain's operations in order to process datasets and make efficient decisions. Recently, deep learning has developed into a significant component of machine learning; it has been used to tackle complicated problems and has demonstrated promising results in a variety of disciplines, including image processing (Wang et al., 2015). It enables computational models composed of several processing layers to acquire knowledge about data representation at various abstraction levels. These techniques greatly improved state-of-the-art object identification, object detection, and speech recognition technologies (LeCun et al., 2015).

Due to the fact that machine learning algorithms are entirely dependent on historical data, and due to the sensitive nature of financial data, protecting user privacy is critical. As a result, publicly available datasets are scarce, limiting the breadth of the study. As a result, any study in this field is limited to a subset of datasets. Additionally, credit card fraud detection systems confront a class imbalance, since the number of fraudulent transactions relative to the hundred thousand regular transactions may be significantly smaller. Addressing class imbalance appropriately is a significant issue for conventional machine learning methods. Deep learning methods are suggested due to their capacity to handle big datasets. One such algorithm is CNN, which is a deep learning approach comparable to artificial neural networks (ANNs). CNN uses the same hidden layer architecture as ANNs. CNNs are widely used in image processing because they conduct feature reduction automatically, making them considerably less prone to overfitting, and therefore training CNN requires less data pre-processing.

**2.1.2.1 Long Short Term Memory**

The Long Short-Term Memory (LSTM) network is a kind of Recurrent Neural Network (RNN). A typical Neutral Network is unable to retain past information and must re-learn the task. In the simplest terms, an RNN is a neural network with memory; it exhibits behavior similar to that of the human brain. Due to the vanishing gradient problem, RNNs have a proclivity towards having short term memory. RNNs lose their capacity to recall early instances in extended sequences, transforming them into short term memory networks. LSTMs solve this short-term memory problem by having a cell state that serves as the network's memory and gates in each step that govern the flow of memory by retaining vital information and rejecting irrelevant information (Malhotra, 2015). LSTM performs somewhat better overall than other algorithms due to its shorter training period.

**2.1.2 Proposed Data Science Libraries and IDE**

Anaconda Navigator is a desktop graphical user interface (GUI) provided with the Anaconda® distribution that enables you to effortlessly launch programs and manage conda packages, environments, and channels without the need to use command-line commands. Navigator may do package searches on Anaconda.org or in a locally installed Anaconda Repository. It runs on Windows, macOS, and Linux. Numerous scientific packages rely on particular versions of other programs to operate. Data scientists frequently utilize various versions of numerous packages and segregate these versions using distinct settings.

Conda is a command-line application that acts as a package manager as well as an environment manager. This enables data scientists to guarantee that each version of each package has all necessary dependencies and functions properly. Navigator is a point-and-click interface for working with packages and environments that eliminates the need to type conda commands in a terminal window. It may be used to locate desired packages, install them in a desired environment, execute the programs, and update them. Anaconda includes a number of applications that are included by default in the Navigator, including the following: i. JupyterLab ii. Jupyter Notebook

iii. Spyder

iv. PyCharm

v. Visual Studio Code

vi. Glueviz

vii. Orange 3 App

viii. RStudio

ix Prompt Anaconda (Windows only)

x. PowerShell Anaconda (Windows only)

**2.2 Review of Related Works**

In 2021, Asha RB and Suresh Kumar KR stated that credit card fraud is prevalent these days due to the increased usage of credit card payment methods. This is due to the advancement of technology and the increase of online transactions, which has resulted in frauds causing significant economic damage. They presented a technique for forecasting the occurrence of fraud by combining several machine learning methods, including support vector machine (SVM), k-nearest neighbor (KNN), and artificial neural network (ANN). Additionally, a distinction of the acquired supervised machine learning and deep learning techniques for detecting fraud and non-fraud transactions was done, yielding an accuracy score of 0.9349, 0.9982, and 0.9992, respectively.

Taha and Malbery (2020) said that advancements in e-commerce and communication technologies have increased the popularity of credit card use as a method of payment, but fraud associated with transactions has also increased. They utilized an improved light gradient boosting machine, in which Bayesian hyper-parameter optimization is combined with the light gradient boosting machine's parameters (LightGBM). They employed two sets of real-world public datasets, comprising both fraudulent and non-fraudulent transactions, in their technique. After evaluating their suggested device using several approaches, they discovered that it outperformed in terms of accuracy. The suggested model achieves 98.40 percent accuracy, 92.88 percent area under the receiver operating characteristic curve (AUC), 97.34 percent precision, and 56.95 percent F1-score.

Sadgali et al. (2019) investigated a novel technique for detecting several types of fraud. The technique, which consists of classification, clustering, and regression, was studied in order to ascertain the contribution and efficacy of each strategy. The authors assert that machine-learning techniques play a critical role in fraud detection and are used to extract and find hidden data. Additionally, hybrid fraud detection systems were the most important due to their ability to combine the characteristics of several conventional methods of detection. However, the majority of hybrid methods failed in real time.

Additionally, Imane Sadgali, Nawal Sael, and Faouzia Benabbou (2019) explain that financial transactions such as internet transactions, credit card transactions, and smartphone transactions have been popular in recent years as a consequence of everyone's preference for digital and cashless transactions. Numerous transactions are conducted, and each one is susceptible to some form of fraud. Numerous researchers investigated, constructed, and built the machine learning model for identifying fraud. They conducted a comparison of all machine-learning algorithms in order to determine which model is the best at detecting fraud in card transactions.

Debachudamani Prusti and Santhnu Kumar Rath created a software that utilized machine learning techniques such as decision trees (DT), k-nearest neighbor (kNN), extreme learning machine (ELM), multilayer perceptron (MLP), and support vector machine (SVM) to determine the accuracy of fraud detection. They presented a model by combining the approaches of DT, SVM, and KNN. They utilized web-based protocols such as simple object access protocol (SOAP) and representational state transfer (REST) to facilitate data interchange across a variety of disparate platforms. A comparison of the outputs of five machine learning algorithms using the accuracy metric. SVM outperformed other algorithms by 81.63 percent, but the hybrid system they suggested had a greater accuracy of 82.58 percent.

Additionally, the writers in (Awoyemi, 2017) examined the overall performance of three ways to financial or personal gain through unlawful or illegal deceit. On credit card data, we used naive Bayes, k-nearest neighbor, and logistic regression. The results revealed that naive Bayes, k-nearest neighbor, and logistic regression all had an accuracy of 97.92, 97.69, and 54.86 percent, respectively. In a comparative research, k-nearest neighbors outperformed naive bayes and logistic regression methods. Additionally, k-nearest neighbor requires training data in order to categorize. As a result, this algorithm is inefficient in detecting anomalous activity because it requires prior training to recognize patterns.

Bhusari and Patil (2016) proposed the Hidden Markov Model as a tool for detecting fraudulent transactions. The Hidden Markov Model uses various transaction ranges such as low, medium, and high as observation symbols. The Hidden Markov model simplified fraud detection systems to the point that they no longer take as long as they used to despite their sophisticated operations. Typically, Hidden Markov Models require training on annotated data. Additionally, this model requires human annotation. As a result, this approach is ineffective in detecting undiscovered patterns of fraud and is ineffective at identifying new fraud patterns. In (Olszewski, 2014; Quah et al., 2008; Zaslavsky et al., 2006), a study suggested a fraud detection model for unsupervised credit card fraud detection that was entirely built on Self-Organizing Map (SOM) neural networks. This approach made use of visualization of user accounts and a type of threshold detection. The pros of this method include the fact that no prior information is required and that the version is continually updated with new credit card transactions; the disadvantage may be the difficulty of identifying fraudulent credit card transactions with maximum accuracy. The SOM method is constrained by a number of constraints, including sluggish processing and a lack of parallelism capacity for large datasets (Fort, 2002).

Jha et al. (2012) employed Logistic Regression to demonstrate the enhanced overall performance of a transaction accumulation method for the purpose of generating suitable derived characteristics for the purpose of detecting credit card fraud. They created a system to improve credit card fraud detection using real-world data/information from credit card transactions. The findings demonstrated the critical nature of classifying information such as product categories, transaction types, and location in order to detect credit card fraud. Additionally, using a Logistic Regression technique to identify fraud is a bad idea because it can only presume a categorical conclusion. Additionally, this set of criteria is criticized for its susceptibility to overfitting (Patil et al., 2018).

Kirkos et al. (2007) identified fraud in monetary transactions by utilizing Bayesian perception networks (BNNs) and decision trees (DTs). A dataset of monetary transactions from seventy-six Greek business entities was used in this study. The dataset included 38 instances of monetary transactions that were determined to be fraudulent by assessors. The BNNs had the highest accuracy (90.3 percent), while the DTs earned 73.6 percent accuracy.

Table 1: Comparison of some related works

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S/n** | **Author** | **Model** | **Accuracy %** | **Drawback** |
| 1 | Asha *et al*. | 1. SVM 2. KNN 3. ANN | 93.49  99.82  99.92 | * SVM is not appropriate for big and complex data. * It requires training by use of annotated data, which means not effective to identify new patterns of fraud. * SVM has lack of result transparency * KNN needs trained dataset to classify as a result, it might not be effective to detect anomaly behavior   No specific drawback |
| 2 | Taha *et al.* | 1. LightGBM 2. RF | 92.40  95.5 | * Was the most accurate approach according to that research study. * RF quickly reaches a point that can’t enhance accuracy. * Need to train data to predict result. |
| 3 | Awoyemi | 1. NB 2. KNN 3. LR | 97.92  97.69  54.86 | * A comparative study shows that KNN can perform better than naïve Bayes and logistic regression. * KNN need trained dataset to classify into fraudulent and legitimate transactions. * It can be used for only categorized data. |
| 4 | Kirkos *et al*. |  | 90.3%  73.6% | * BNN achieved the best result * DT also achieved an accurate result |

**CHAPTER THREE**

**RESEARCH METHODOLOGY**

**3.1 Design Consideration**

The primary algorithm to be utilized in this research is Long Short Term Memory (LSTM), which will be compared to Autoencoder in this part. The design technique also includes a comparison of LSTM to another deep learning algorithm for cross assessment. Long Short Term Memory networks, or LSTMs for short, are a kind of RNN capable of learning long-term dependencies. They were pioneered by Hochreiter & Schmidhuber (1997) and improved and popularized by a number of individuals in subsequent work. They perform exceptionally well on a wide variety of sequence modeling tasks and have gained widespread use. LSTMs are purpose-built to eliminate the problem of long-term reliance. Their default mode of operation is to retain information for extended periods of time.

The recurrent neural network is a subtype of neural network that considers not only the current input but also the past input.

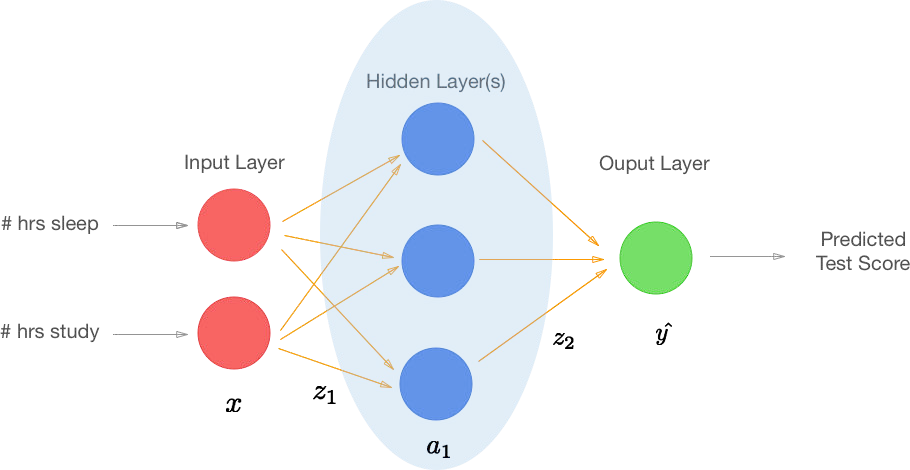


Figure 1: An example of Recurrent neural network (Mikami *et. al* 2016)

The design depicts a recurrent neural network in its entirety. The green block labeled A is a well-known basic feed forward neural network. The right hand side of the equality sign in the figure depicts the network for each time step, i.e. at t=0, input X0 enters the network to generate H0, the following time step, the input is X1, but there is an extra input from the previous time step from block A. Thus, the neural network considers not only the present input but also the context of past inputs.

With this structure, the RNN model begins to worry about the past and the future. We can refer to this as memory since the recurrent units store the previous values. Now, we may explore the true significance of context in data.

LSTMs also have a similar topology, however their internals differ from the RNN's single tanh (activation) layer. An LSTM block contains four layers that interact with one another.

The LSTM network's structure is composed of memory blocks (cells) with multiple states and gates. The cell state is the primary link in the information flow chain. It enables unmodified information to go ahead. The forget gate (ft) regulates which data must be deleted or retained. The sigmoid function is used to transfer data from the previous hidden state (ht-1) together with data from the current input (Xt). The sigmoid function (r) determines values between 0 and 1; values closer to 0 indicate forgetting, while values closer to 1 indicate keeping. Additionally, the cell state vector Ct-1 specifies which items will be forgotten.

ft = (W*f*  [ht-1, Xt] + b*f* ) (1)

Let bf and Wf indicate the forget gate's bias and weight matrices, respectively (ft). The input gate (It) selects which data from the current input (Xt) should be added and also updates the cell state. This gate utilized the tanh function (Nt) to convert the current input and hidden state to values between -1 and 1 in order to help with network regulation. Additionally, it is applied to the old cell-stated (Ct-1) memory at time t -1 to generate new cell-stated (Ct) at time t.

**3.2 Architectural Framework**

The overall framework of the proposed intelligent approach for credit card fraud detection is illustrated in below;

Pre processing

Historical data

Structured

Data

Training set

Model training / building

Deep learning algorithm (LSTM & Auto-encoder)

Deploy model

Test model prediction

prediction

Figure 2: Overall framework of the proposed Credit card fraud detection system Using LTSM and Auto-Encoder Algorithm.

The proposed model for credit card fraud detection consists of four major steps, which are explained in the following subsections below. The experiment was performed using an Intel Core i7 processor with 8GB RAM. The proposed approach and other machine learning techniques were implemented and tested using Python.

**3.2.1** **Fraud Dataset**

To begin, I obtained the dataset from Kaggle, a data analysis platform that makes datasets available for free.

The statistics contain credit card transactions performed by European cardholders in September 2013. This dataset contains transactions that happened over the course of two days and contains 492 frauds out of 284,807 total transactions. The dataset is very asymmetrical; positive transactions (frauds) account for 0.172 percent of all transactions. Additionally, this dataset has 31 variables, including time, quantity, class, and other critical characteristics. Numerous researchers have utilized it as a benchmark dataset (Carcillo et al., 2018; Dal Pozzolo et al., 2014).

**3.2.2 Data preprocessing**

The experimental process begins with data pre-processing. In this stage, data sets are thoroughly examined by manually querying them and performing statistical procedures. The goal of data pre-processing is to offer a refined input to classifiers in order to obtain the best output possible. Missing values, categorical features, changing scale, and a high degree of dimensionality can all have an effect on the classifier's performance. Data exploration, data scaling, and test-train split are the two pre-processing approaches used in this study.

Three processes should be performed before to applying the model to the dataset: data validation, normalization, and division.

a) Validation of Data

This phase is used to validate the data contained inside the dataset, such as a negative time value, empty values, or a negative quantity.

b) Normalization

To provide an accurate result, the model rescales the variables between -1 and 1. This step is required to convert the numeric column values in the dataset to a common scale without distorting the value ranges or erasing data.

c) Division of the dataset's sample

It is critical to divide the data samples into training and testing in order to obtain a meaningful assessment of performance. The suggested model uses 70% of the dataset for training and 30% for testing.

Table 2: LSTM parameters and performance measures.

|  |  |
| --- | --- |
| Parameters | Description |
| Optimizer | Parameter that work to enhance the performance. It has different types such as, Adam, Adagrad… etc |
| Metric | Measure the performance like accuracy, execution time etc |
| Batch Size | Refers to the number of windows of data we are passing at once |
| Epochs | Refers to the number of iteration s (forward and back propagation) model needs to make |
|  |  |

Table 3: Optimizers of LSTM model

|  |  |
| --- | --- |
| Parameters | Description |
| Adam | Adaptive Moment Estimation (ADAM) optimizes the computation of learning rates for each parameter in deep learning artificial neural network methods. It accomplishes this aim by using the gradient's first and second moments. Additionally, ADAM is beneficial for situations involving a huge volume of data (Ruder 2016). |
| Adagrad | Adaptive Gradient (Adagrad) is a technique for optimizing using gradients. Adagrad allows adaptive learning rates and automatically adjusts the settings. It conducts more frequent updates. Due to this property, it is advantageous in issues with sparse data (Ruder 2016). |
|  |  |
|  |  |

The model uses several parameters called hyperparameters that work to improve the results. To measure the performance of the proposed model, four parameters namely optimizer, batch size, epochs, and matrix are used and described in Table 2. There are a considerable number of optimizers that could make the difference between models, some of these optimizers are shown in Table 3.

**3.2.3 Data Scaling and Standardization**

Deep learning algorithms perform poorly when the input characteristics are not reasonably comparable in size. Scaling and standardization approaches reduce the scale of the characteristics to a near-identical value in order to make the input more understandable to the classifier. The StandardScaler class from the sklearn pre-processing package in Python is utilized in this investigation. The standard scaler converts each feature in the dataset to have a mean of zero and a standard deviation of one. After applying the conventional scaler to the dataset, the mean value for the same two characteristics described previously was reset to zero.

**3.3 Autoencoder**: An autoencoder in its simplest form is a feedforward, non-recurrent neural network resembling single layer perceptions that participate in multilayer perceptions (MLP) – utilizing an input layer and an output layer coupled through one or more hidden layers. The output layer is identical to the input layer in terms of nodes (neurons). Rather than forecasting a target value Y given inputs X, its aim is to rebuild its inputs (minimizing the difference between the input and the output). Thus, autoencoders are models of unsupervised learning. (Learning does not require labeled inputs.)

An autoencoder is composed of two components: an encoder and a decoder, which may be described as transitions (phi) and (psi), respectively, such that:

(2)

(3)

(4)

In the simplest case, given one hidden layer, the encoder stage of an autoencoder takes the input

x ∈ R d = X {\displaystyle \mathbf {x} \in \mathbb {R} ^{d}={\mathcal {X}}}x ϵ ℝd = 𝝌 and maps it to 𝗵 ϵ ℝp = 𝓕 :

𝗵 = (**W**x + b) (5)

This image h {\displaystyle \mathbf {h} } 𝗵 is usually referred to as *code*, *latent variables*, or *latent representation*. Here, σ {\displaystyle \sigma } is an element-wise activation function such as a sigmoid function or a rectified linear unit. W {\displaystyle \mathbf {W} } is a **W** weight matrix and b b {\displaystyle \mathbf {b} } is a bias vector. Weights and biases are usually initialized randomly, and then updated iteratively during training through backpropagation. After that, the decoder stage of the autoencoder maps 𝗵 h {\displaystyle \mathbf {h} } to the reconstruction **x ′ {\displaystyle \mathbf {x'} } x**i of the same shape as **x {\displaystyle \mathbf {x} } x**:

xi = (**Wi** 𝗵 + bi), (6)

where , Wi and bi σ ′ , W ′ ,  and  b ′ {\displaystyle \mathbf {\sigma '} ,\mathbf {W'} ,{\text{ and }}\mathbf {b'} } for the decoder may be unrelated to the corresponding , W, b σ , W ,  and  b {\displaystyle \mathbf {\sigma } ,\mathbf {W} ,{\text{ and }}\mathbf {b} } for the encoder.

Autoencoders are trained to minimize reconstruction errors (such as squared errors), often referred to as the "loss":

L(x, x') = || x-x ||2 = ||x - ||2  (7)

where x {\displaystyle \mathbf {x} } x is usually averaged over some input training set.

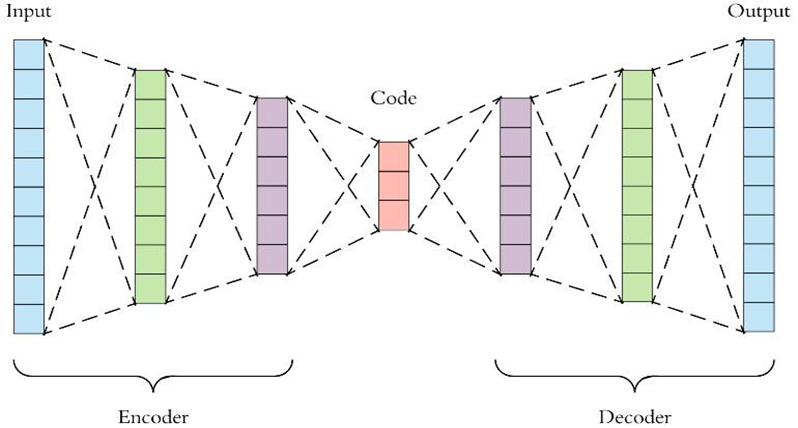


Figure 3: Autoencoder Architecture (Mikami *et. al* 2016)

**3.4 Evaluation Metrics**

The model configuration is used to demonstrate convincingly that the LSTM model is effective in detecting abnormalities in the finance sector. The various activities, including as pre-processing and model selection, are programmed using functions from the Python Sklearn package. The LSTM model is implemented using the Keras package. The purpose of this study is to determine the accuracy, loss rates, and execution time associated with financial detection fraud. Additionally, it demonstrates that the LSTM model is capable of detecting both known and undiscovered fraud tendencies.

1. **Accuracy**

The accuracy metric is a statistical measurement that indicates how well a model predicts (Ahmed & Bahador, 2018). Calculating the accuracy serves the purpose of determining the model's efficiency. The following equation illustrates how the accuracy measure is determined;

Where TP denotes true positive, suggesting that a portion of suspicious transactions were properly classified as suspicious, and TN denotes true negative, indicating that a fraction of normal transactions were correctly classified as normal. The term "false positive" refers to the percentage of non-suspicious transactions that are erroneously categorized as suspicious, whereas "false negative" refers to the percentage of suspicious transactions that are incorrectly classed as normal transactions (Sorournejad et al., 2016).

1. **Loss rate**

The loss rate is a function that quantifies the difference between actual and projected output during training in order to accelerate the learning process (Ahmed & Bahador, 2018). Additionally, the loss rate is utilized to evaluate the model's performance and error minimization. Calculate the loss rate using the equation below;

Where Y denote actual output and YPred denote predict output.

1. **Execution time**

The execution time is defined as the time required for the model to complete the task. Calculating the time required for execution enables us to determine how long the model takes to detect frauds while also ensuring that the model accomplishes its objective effectively. The following equation illustrates how execution time is determined.

Time = Timeend – Timestart

**3.5 Psuedocodes for LSTM and Auto-Encoder**

Table 4: Auto-Encoder Pseudo-coding.

|  |  |
| --- | --- |
| Steps | Processes |
| Step 1: Prepare the input data | Input Matrix X // input dataset  Parameter of the matrix // parameter (w, bx, bh)  where: w : Weight between layers, bx Encoder’s parameters , bh Decoder’s Parameters |
| Step 2: initial  Variables | h ← null // vector for hidden layer  X ← null // Reconstructed x  L ← null // vector for Loss Function  l ← batch number  i ← 0 |
| Step 3: loop  statement | While i < l do  // Encoder function maps an input X to hidden representation h:  𝗵 = (**W**x + b)  Decoder function maps hidden representation h back to a  Reconstruction X) :\*/  xi = (**Wi** 𝗵 + bi), /\*For nonlinear reconstruction, the reconstruction loss is generally  from cross-entropy :\*/  L(x, x') = || x-x ||2 = ||x - ||2  /\* For linear reconstruction, the reconstruction loss is generally from the squared error:\*/  L End while  Return θ |
| Step 4: output | θ ← <null matrix>//objective function  /\*Training an autoencoder involves finding parameters =  (W,bx , bh) that minimize the reconstruction loss on the given  dataset X and the objective function\*/ |

Table 5: LSTM Pseudo-coding

|  |  |
| --- | --- |
| Steps | Processes |
| Step 1 | Set *ipunits*, *lstmunits*, *opunits* and optimizer to define LSTM Network (L) |
| Step 2 | Normalize the dataset(Di) into values from 0 to 1 using (4) |
| Step 3 | Select training window size (tw) and organize Di accordingly |
| Step 4 | for n epochs and batch size do  Train the Network (L)  end for |
| Step 5 | Run Predictions using L |
| Step 6 | Calculate the loss function using (9) |

**CHAPTER FOUR**

**IMPLEMENTATION AND EVALUATION**

**4.1 Hardware & Software Requirements**

The Credit Card Fraud Detection System is compatible with any microcomputer setup that meets the following requirements:

1. A hard disk of at least 100GB
2. Core i3, i5 or i7 processor
3. 8GB RAM memory
4. Pentium M 2.4 GHz
5. Windows 8.1 or 10 operating system.

**4.2 Choice of Programming Environment**

Anaconda Jupyter is Integrated Development Environment used for their implementation and development of the credit card fraud detection system.

**4.2.1 Language Justification**

Python programming language was used in the development of the system. The choice of using the language was based on the recommendation and research made as regards deep learning analysis and implementation.

**4.3 System Testing**

The project implementation was tested on various computer with different configuration to determine the accuracy, speed and efficiency of the system. System configuration with i3, i5 and i7 core processor were used in the test running of the program.

**4.3.1 Test Data**

Dataset used in the testing of the program was downloaded from Kaggle, a database repository for dataset for data mining and analysis. An American dataset for Credit card fraud was downloaded and utilized for the implementation.

**4.3.2 Performance Evaluation**

Accuracy, Loss rate and Execution time are the major metrics used in determining the performance evaluation of the system. The values gotten from the metrics listed above varies because of the different testing environment used.

**4.4 Limitation of the System**

A C.P.U and Intel graphics based system was used for the testing, running and implementation of the system which hinders the EPOCH running rate of the program. A GPU-based system is highly recommended for faster and accurate result of the program.

**4.5 Results and Discussion**

**4.5.1 Results**

LSTM and Autoencoders are the two Deep learning techniques used for the implementation of the program. Long Short-Term Memory (LSTM) and Autoencoders gives more finite and accurate results compared to SVM, RandomForest algorithm which are Machine learning based techniques.

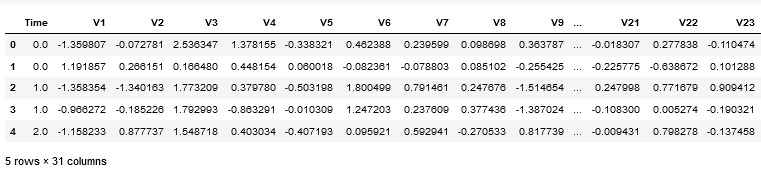


Figure 15: Interpreted table of the credit card dataset

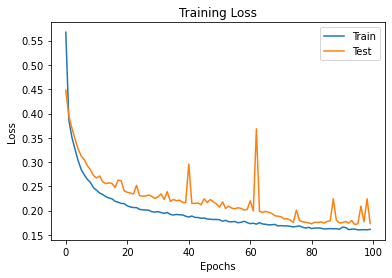
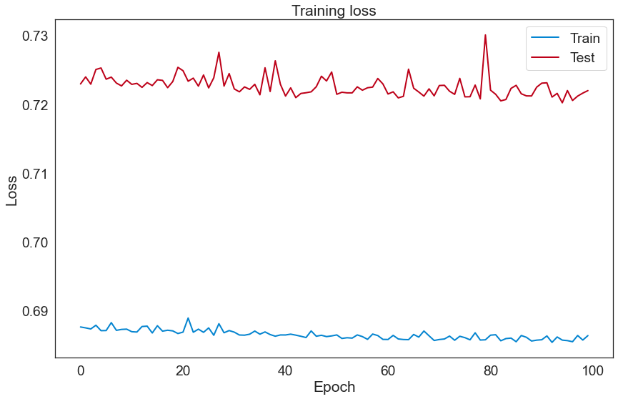


Figure 16: Model Loss vs Epoch Graph for Train and Train data

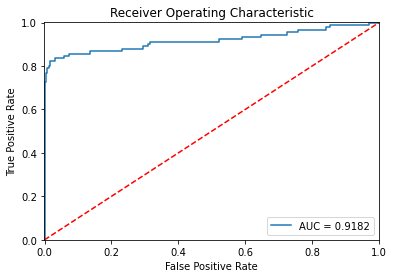
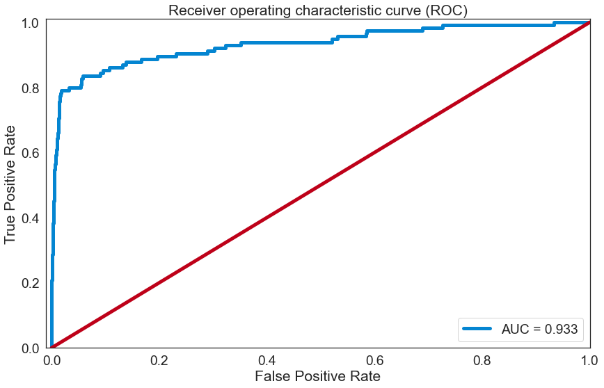


Figure 17: ROC graph for LSTM and Autoencoder

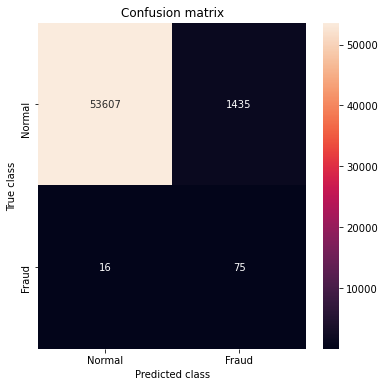
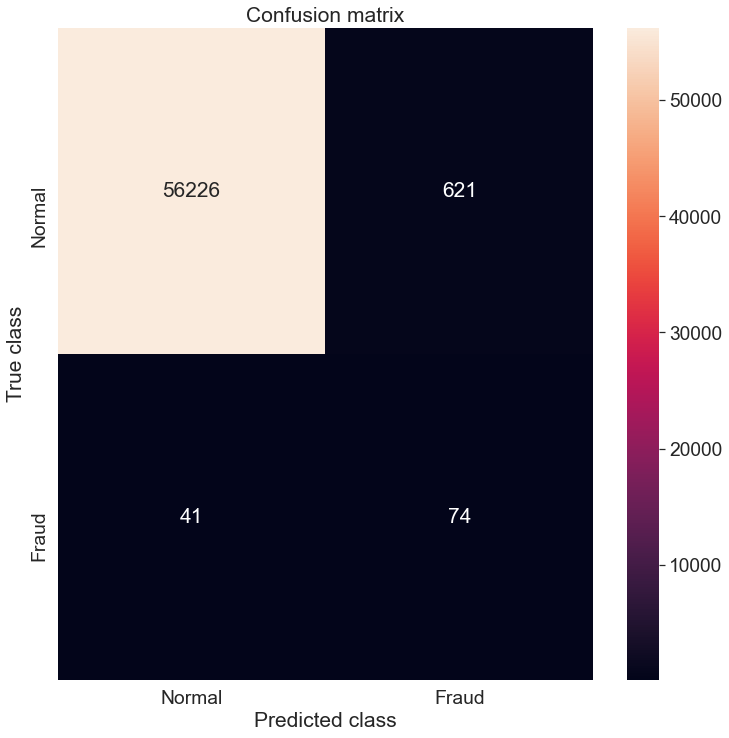


Figure 18: Confusion Matrix graph for LSTM and Autoencoder

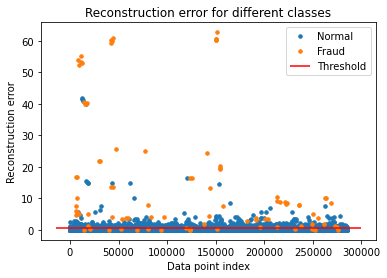


Figure 19: Reconstruction Error for LSTM and Auto-Encoder

Table 1: Result of experiment with different optimizers

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Optimizer | Training Accuracy (%) | Validation Accuracy (%) | Loss (%) | Number of Iterations |
| Adam | 99.98% | 99.97 | 0.32 | 100 |
| Adagrad | 91.99% | 93% | 0.65 | 100 |

Table 2: Comparison of LSTM and auto-encoder

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Model | Optimizer | Training Accuracy (%) | Validation Accuracy (%) | Loss (%) | Number of Iterations |
| LSTM | Adam | 99.98% | 99.97 | 0.32 | 100 |
| Auto-Encoder | Adam | 70.24% | 70.20% | 0.69 | 100 |

**4.5.2 Discussion**

As shown in Table 1, comparison of the model results in 91.99 percent accuracy and 0.65 percent loss rate in 429s utilizing the Adagrad optimizer, 100 iterations, and three layers for encoding and decoding. To assess the suggested model's accuracy and speed, it is run under several conditions, including different optimizers, varied layer counts, and iterations. Numerous optimizers, including as Adam and Adagrad, were employed in the assessment; each optimizer produces a unique result. The purpose of the implementation is to ascertain which optimizer achieves the highest performance. The results of several trials indicated that Adam optimizer achieved the highest level of accuracy, as shown in Table 1. It is well-known that the Adam optimizer is capable of dealing with enormous amounts of data, does not require a lot of memory, and is computationally efficient (Kingma & Ba, 2015). As a consequence of the positive findings reported in Table 1, the Adam optimizer was further examined on several layers and a varied number of iterations.

Based on the use of LSTM cells, the model easily and successfully predicts credit card fraud detection. Although this model supports a huge number of optimizers, picking the optimal optimizer leads in a substantial increase in outcomes. The number of layers and iterations used in the implementation had little effect on the outcomes. Additionally, the findings demonstrate that the number of iterations and the number of layers have an effect on the time required. The results established a substantial connection between accuracy and iteration count. Additionally, there is an inverse connection between loss rates and iteration count. There is a negative link between accuracy and loss rates, as illustrated in Figure 16, as when accuracy improves, loss rates drop.

**4.5.2.1 Comparison of the Models**

Comparison of the two model; LSTM and Auto-Encoder is targeted to determine the efficiency of each models. Adam optimizer show high accuracy, low loss accuracy and speed over other optimizers in Table 2. The LSTM model gives the most accuracy result compare to Auto-Encoder. LSTM model gives 99.98% accuracy, 0.32% loss rate in 500s then Auto-Encoder gives 70.24% accuracy, 0.69% loss rate on 395s. This shows that LSTM can be used for big and complex data for data mining and analysis.

**4.5.2.2 Comparison with existing machine learning based techniques**

Table 3: Comparison with existing machine learning based techniques.

|  |  |  |
| --- | --- | --- |
| Model | Accuracy | Comments |
| SVM | 99.87 | * SVM required training by use of annotated data which means not effective to identify new patterns of fraud * SVM has lack of results transparency. |
| Random Forest | 97.95% | * RF can achieve good results with small number of data |
| Logistic Regression | 99.89 | * Its weakness to over-fitting * Logistic Regression it can expect only a definite outcome. |
| LSTM | 99.98% |  |

Table 3 shows performance comparison of the proposed model with other machine learning techniques based on the accuracy metrics. The proposed model gives the highest accuracy of 99.98% , while Random Forest gives the low accuracy of 97.95%. Though the machine learning technique yield high results but they are unable to process big data and also detects new pattern and features. Nevertheless, deep learning-based methods can learn even through complex data patterns and dynamically adapt with new patterns of frauds.

**CHAPTER FIVE**

**CONCLUSION, RECOMMENDATION AND FUTURE WORK**

**5.1 Conclusion**

The credit card fraud detection model using Autoencoder and LSTM were successfully implemented and ROC curve, Reconstruction error and Training model loss were plotted. The result obtained in case of the LSTM is higher than the others. This fraud detection model is trained using the European credit card fraud data. The data availability in case of credit card fraud detection is scarce. Since model learn from data and not from labels it can be transferred to other dataset too.

Two optimizers were used; Adam and Adagrad for result comparison, training accuracy, validation accuracy and loss model evaluation. Adam optimizer gives the best and reliable result accuracy. Accuracy value of 99.98% is derived using LSTM while 70.24% of accuracy was derived using Auto-Encoder model.

**5.2 Recommendation**

The current advantages Deep learning provides makes it essential to include its techniques into the proposed credit card fraud detection system. This approach proposed can be also adopted to other domain in fraud detection like Retail Banking, Point of Sale (POS) system, Insurance system and so on.

**5.3 Contribution to Knowledge**

This research contributes by proposing a fraud detection model based on the deep learning techniques Long Short – Term Memory (LSTM) and Autoencoder. Comparison of the old model's performance assessment metrics to the newly adopted model's performance evaluation metrics.

**5.4 Future Work**

I look forward to introducing new deep learning model and technique to solve more complex fraud detection problems. And also benchmarking existing model; LSTM and Auto-Encoder with the new proposed deep learning technique to be adopted.

**REFERENCES**

Ahmed, W., & Bahador, M. (2018). *The accuracy of the LSTM model for predicting the S&P 500 index and the difference between prediction and backtesting. (Degree thesis)*. Stockholm, Sweden: KTH Royal Institute of Technology, School Of Electrical Engineering and Computer Science.

Andrea, D. P., 2015. *Adaptive Machine Learning for Credit Card Fraud Detection*. PhD Thesis, Department of Computer Science, Université Libre de Bruxelles.

Asha RB, Suresh Kumar KR, 2021. *Credit card fraud detection using artificial neural network*, Global Transitions Proceedings, Volume 2, Issue 1, 2021, Pages 35-41.

Awoyemi, J., 2017. *Credit card fraud detection using machine learning techniques: comparative analysis*. Conference on Computing Networking and Informatics (ICCNI), International, IEEE.

Bhattacharyya, S., Jha, S., Tharakunnel, K., & Westland, J.C., “*Data mining for credit card fraud: A comparative study*,” Decis. Support Syst., vol. 50, no. 3, pp. 602–613, 2011

Bhusari, V., & Patil, S. (2016). *Study of hidden markov model in credit card fraudulent detection.* ld Conference on Futuristic Trends in Research and Innovation for Social Welfare (Startup Conclave)., 2016 Wor, IEEE. <https://doi.org/10.1109/STARTUP.2016.7583942>

Carcillo, F., Dal Pozzolo, A., Le Borgne, Y.-A., Caelen, O., Mazzer, Y., & Bontempi, G. (2018). Scarff: *A scalable framework for streaming credit card fraud detection with spark*. Information Fusion, 41, 182–194. https://doi.org/10.1016/j.inffus.2017.09.005

Dal Pozzolo, A., Boracchi, G., Caelen, O., Alippi, C., & Bontempi, G. “*Credit card fraud detection: a realistic modeling and a novel learning strategy*”. IEEE Trans. Neural Netw. Learn. Syst. vol 29,no 8, pp 1–14 .3784-3797,Sep.2017.

Dal Pozzolo, A., Caelen, O., Le Borgne, Y.-A., Waterschoot, S., & Bontempi, G. (2014). *Learned lessons in credit card fraud detection from a practitioner perspective*. Expert Systems with Applications, 41(10), 4915–4928. <https://doi.org/10.1016/j.eswa.2014.02.026>

Fort, J.-C. (2002). *Advantages and drawbacks of the Batch Kohonen algorithm*. ESANN. journal of applied security research 17

Jha, S., Guillen, M., & Christopher Westland, J., 2012. *Employing transaction aggregation strategy to detect credit card fraud*. Expert Systems with Applications, 39(16), 12650–12657.

Kirkos E., C. Spathis, and Y. Manolopoulos, 2007 “*Data mining techniques for the detection of fraudulent financial statements*,” Expert Syst. Appl., vol. 32, no. 4, pp. 995-1003.

Lebichot, Bertrand, Yann-Aël Le Borgne, Liyun He-Guelton, Frédéric Oblé, and Gianluca Bontempi, 2019. “*Deep-Learning Domain Adaptation Techniques for Credit Cards Fraud Detection.*” in Proc. INNS Big Data and Deep Learning conference. Genoa, Italy, 2019. pp. 78-88.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436–444. <https://doi.org/10.1038/nature14539>

M. Carminati, R. Caron, F. Maggi, I. Epifani, and S. Zanero, BankSealer: “*A Decision Support System for Online Banking Fraud Analysis and Investigation*” Computers & Security, vol 53, no 1, pp. 175-86.Sep.2015.

Mikami A., “*Long Short-Term Memory Recurrent Neural Network Architectures for Generating Music and Japanese Lyrics*” Department of Computer science, Boston College. 2016.

Odumuyiwa, T., Kennedy, M., Norman, A., Holloway, M., Suffeld, F., Forrest, H. and Dicks, H. ,2019. *Improving Access to Social Care Services Following Acquired Brain Injury: A Needs Analysis*. Journal of Long-Term Care, 164 -175.

Olszewski D., “*Fraud detection using self-organizing map visualizing the user profiles*” Knowledge.-Based Systems., vol. 70, pp. 324– 334.

Patil, S., Nemade, V., & Soni, P. K. (2018). *Predictive modelling for credit card fraud detection using data analytics*. Procedia Computer Science, 132, 385–395. [https://doi.org/10. 1016/j.procs.2018.05.199](https://doi.org/10.%201016/j.procs.2018.05.199)

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

Quah, J.T., & Sriganesh, M., “*Real-time credit card fraud detection using computational intelligence*,” Expert Syst. Appl., vol. 35, no. 4, pp. 1721–1732, 2008. ­

R. Bolton and D. Hand, “*Statistical fraud detection: A review*” Stat. Sci., vol. 17, no. 3, pp. 235–249, Aug.2002.

Ravisankar, P., Ravi, V., Rao, G.R., & Bose, I. “*Detection of financial statement fraud and feature selection using data mining techniques*”' Decis. Support Syst., vol. 50, no. 2, pp. 491-500, Jan.2011.

Sadgali, I., Sael, N., and Benabbou, F. 2019. *Performance of machine learning techniques in the detection of financial frauds.* Procedia Computer Science, 148, 45–54.

Sorournejad, S., Zojaji, Z., Atani, R. E., & Monadjemi, A. H. (2016). *A survey of credit card fraud detection techniques*: Data and technique oriented perspective. arXiv:1611.06439.

Taha, A.A., Malebary, S.J., *An intelligent approach to credit card fraud detection using an optimized light gradient boosting machine*, IEEE Access, 8 (2020), pp. 25579-25587, 10.1109/ACCESS.2020.2971354

Thanh Thi N, Hammad Tahir, Mohamed Abdelrazek, Ali Babar, 2020. *Deep Learning Methods for Credit Card Fraud Detection*. <https://arxiv.org/abs/2012.03754v1>

Tripathi K and Pavaskar M, 2012. *Survey on credit card fraud detection methods* International Journal of Emerging Technology and Advanced Engineering 2, 721-726

Wang, L.; Liu, T.; Wang, G.; Chan, K. L.; and Yang, Q. “ *Video tracking using learned hierarchical features*” IEEE Trans. Image Process, vol. 24, no 4, pp.1424-1435,2015.

West, J., Bhattacharya, M. (2016). *Intelligent financial fraud detection: a comprehensive review. Computers & Security*, 57, 47 –66. <https://doi.org/10.1016/j.cose.2015.09.005>

Yomas, J., & Kiran, C. N. (2018). *Critical analysis on the evolution in the e-payment system, security risk, threats and vulnerability*. Communications on Applied Electronics, 7(23), 21–29. <https://doi.org/10.5120/cae2018652800>

Zaslavsky, V., and Strizhak, A., “*Credit card fraud detection using self-organizing maps*” Inf. Secur., vol. 18, p. 48, 2006.

**APPENDIX**

import pandas as pd

import numpy as np

from scipy import stats

import tensorflow as tf

import matplotlib.pyplot as plt

import seaborn as sns

import pickle

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import confusion\_matrix, precision\_recall\_curve

from sklearn.metrics import recall\_score, classification\_report, auc, roc\_curve

from sklearn.metrics import precision\_recall\_fscore\_support, f1\_score

from sklearn.preprocessing import StandardScaler

from pylab import rcParams

from keras.models import Model, load\_model

from keras.layers import Input, Dense

from keras.callbacks import ModelCheckpoint, TensorBoard

from keras import regularizers

#set random seed and percentage of test data

RANDOM\_SEED = 314 #used to help randomly select the data points

TEST\_PCT = 0.2 # 20% of the data

#set up Colour style

rcParams['figure.figsize'] = 14, 8.7 # Golden Mean

LABELS = ["Normal","Fraud"]

col\_list = ["cerulean","scarlet"]

sns.set(style='white', font\_scale=1.75, palette=sns.xkcd\_palette(col\_list))

#Add CSV File

df = pd.read\_csv("C:/Users/ACER/Desktop/Ayo programs/tm/creditcard.csv") #Make sure the CSV file is located in the same folder with this Code!

df.head(n=5) #just to check you imported the dataset properly

#This section is used to confirm the data imported

df.shape #secondary check on the size of the dataframe

#check to see if any values are null, which there are not

df.isnull().values.any() #THe output should be False

#This section will give the output for Normal and Fraudulent rows of Data

pd.value\_counts(df['Class'], sort = True) #class comparison 0=Normal 1=Fraud

#Balance of Data Visualization

#Visual confirmation of unbalanced Data in the Dataset

count\_classes = pd.value\_counts(df['Class'], sort = True)

count\_classes.plot(kind = 'bar', rot=0)

plt.xticks(range(2), LABELS)

plt.title("Frequency by observation number")

plt.xlabel("Class")

plt.ylabel("Number of Observations");

#Summary Statistics of the Transaction Amount Data

normal\_df = df[df.Class == 0] #save normal\_df observations into a separate df

fraud\_df = df[df.Class == 1] #do the same for frauds

#Description of Differences btw the Normal and Fraud data

normal\_df.Amount.describe() #For Normal

fraud\_df.Amount.describe() #For Fraud

#Visual Exploration of the Transaction Amount Data

#plot of high value transactions

bins = np.linspace(200, 2500, 100)

plt.hist(normal\_df.Amount, bins, alpha=1, density=True, label='Normal')

plt.hist(fraud\_df.Amount, bins, alpha=0.6, density=True, label='Fraud')

plt.legend(loc='upper right')

plt.title("Amount by percentage of transactions (transactions \$200+)")

plt.xlabel("Transaction amount (USD)")

plt.ylabel("Percentage of transactions (%)");

plt.show()

#Creating the Model

#Autoencoder Layer Structure and Parameters

nb\_epoch = 100

batch\_size = 128

input\_dim = train\_x.shape[1] #num of columns, 30

encoding\_dim = 14

hidden\_dim = int(encoding\_dim / 2) #i.e. 7

learning\_rate = 1e-7

input\_layer = Input(shape=(input\_dim, ))

encoder = Dense(encoding\_dim, activation="tanh", activity\_regularizer=regularizers.l1(learning\_rate))(input\_layer)

encoder = Dense(hidden\_dim, activation="relu")(encoder)

decoder = Dense(hidden\_dim, activation='tanh')(encoder)

decoder = Dense(input\_dim, activation='relu')(decoder)

autoencoder = Model(inputs=input\_layer, outputs=decoder)

#Confusion Matrix

pred\_y = [1 if e > threshold\_fixed else 0 for e in error\_df.Reconstruction\_error.values]

conf\_matrix = confusion\_matrix(error\_df.True\_class, pred\_y)

plt.figure(figsize=(12, 12))

sns.heatmap(conf\_matrix, xticklabels=LABELS, yticklabels=LABELS, annot=True, fmt="d");

plt.title("Confusion matrix")

plt.ylabel('True class')

plt.xlabel('Predicted class')

plt.show()