FRAUD DETECTION IN ONLINE TRANSACTIONS USING DEEP LEARNING TECHNIQUES

Thesis Submitted For the Award of the Degree of

DOCTOR OF PHILOSOPHY

In

Computer Science & Engineering

By

Kanika

Registration No. 41800913

Supervised By

Dr. Jimmy Singla

LOVELY PROFESSIONAL UNIVERSITY
PUNJAB
2022

DECLARATION

I, Kanika, declare that this thesis titled “Fraud Detection in Online Transactions using Deep Learning Techniques” has been composed by myself and contains original work of my own execution.

I further declare that to the best of my knowledge the thesis does not contain any part of any work which has been submitted for the award of any degree either by this University or any other university/ deemed university without proper citation This thesis has been written under the supervision of Associate Professor Dr. Jimmy Singla.

Kanika
Reg. No. 41800913
School of Computer Science and Engineering,
Lovely Professional University,
Phagwara, Punjab-144411, India
Date:

CERTIFICATE

This is to certify that thesis entitled "**Fraud Detection in Online Transactions using Deep Learning Techniques**" is a piece of research work done by Ms. Kanika Reg. No. 41800913 under my supervision for the degree of Doctor of Philosophy in School of Computer Science and Engineering of Lovely Professional University, Punjab, India.

To the best of my knowledge and belief, the thesis

I. embodies the work of candidate herself.
II. has duly been completed.
III. fulfils the requirement of ordinance related to Ph.D. degree of the University.
IV. is up to the standard both in respect of content and language for being referred to the examiner.
V. has not been submitted to any other university or institution for the award of any degree or diploma.

Dr. Jimmy Singla
Associate Professor
School of Computer Science and Engineering,
Lovely Professional University,
Phagwara, Punjab-144411, India
Date:

Abstract

Fraud Detection in Online Transactions using Deep Learning Techniques

With the development of information technology, most of the modern commerce is depending upon the online banking and cashless payments. Online transaction services have provided great convenience to the customers. However, fraud cases are also increasing day by day. The traditional rule-based systems are incapable to deal with the changing fraud patterns. Also, the frequency of fraud transactions is generally less as compared to the genuine transactions occurring at a time. Thus, detecting frauds among large number of transactions occurring simultaneously in real time is also very crucial. Hence, there is an urgent need for an intelligent system which is capable enough to detect the online transactions fraud from large pool of transactions in real time in an accurate and fast manner. In this thesis, i) a deep learning-based fraud detection system has been proposed to detect online fraudulent transactions from imbalanced datasets. ii) Various Algorithm-level techniques have been used to handle class imbalance problem of datasets at the cost of misclassification of few genuine transactions which have performed better in terms of fraud detection rate than data-level and hybrid methods. iii) Thresholding has been performed to optimize the decision threshold of the deep learning model to maximize the fraud detection rate of the system. Various thresholding methods selected for optimizing decision threshold have been assessed to demonstrate that the selection of right thresholding method with deep learning yields better results. iv) Finally, the comparison of proposed deep learning-based fraud detection system has been done with machine learning based fraud detection systems.

Acknowledgment

Foremost, I would like to express my deep and sincere gratitude to my supervisor Associate Professor Dr. Jimmy Singla for his inspiration, patience, and encouragement. He introduced me to the field of Machine Learning and provided me with a lot of good ideas and advice. I would have never completed this thesis without his help. I am deeply grateful to him for his suggestions and guidance throughout my Ph.D. study. His great ideas and unfailing support were essential to finish this thesis.

My sincere thanks also go to the Ph.D. committee members and the examiners of this thesis for the insightful comments and hard questions throughout my Ph.D. study. If I was able to arrive at the Ph.D. studies, it is because I have great parents that always believe in what I am doing. I wish to convey special thanks to my family for their endless support and understanding.

Finally, I would like to thank my friends from the School of Computer Science & Engineering, LPU.

Kanika

Table of Contents

List of Abbreviations

Abbreviation	Explanation
UPI	Unified Payments Interface
OA	Online Auction
P2P	Peer-to-Peer
PID	Personally Identifiable Data
FTP	File Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
DL	Deep Learning
DNN	Deep Neural Network
CNN	Convolutional Neural Network
ML	Machine Learning
KNN	K Nearest Neighbor
DT	Decision Tree
LR	Logistic Regression
RF	Random Forest
LGBM	Light Gradient Boosting Machine
MLP	Multi-Layer Perceptron
RUS	Random Undersampling
ROS	Random Oversampling
SMOTE	Synthetic Minority Over-sampling Technique
FL	Focal Loss
CEL	Cross Entropy Loss
W-CEL	Weighted-Cross Entropy Loss
W-FL	Weighted-Focal Loss
RFL	Reduced Focal Loss
WH-RFL	Weighted-Hard Reduced Focal Loss
ROC	Receiver Operating Characteristic
TPR	True Positive Rate
FPR	False Positive Rate

TNR	True Negative Rate
G-Mean	Geometric-Mean
AUROC	Area Under Receiver Operating Characteristics
LSTM	Long Short-Term Memory
GAN	Generative Adversarial Network
CDR	Customer Details Record
SVM	Support Vector Machine
GBC	Gradient Boosting Classifier
DCNN	Deep Convolutional Neural Network
AE	Autoencoder
RBM	Restricted Boltzmann Machine
BCE	Binary Cross Entropy
RNN	Recurrent Neural Network
NN	Neural Network
PCA	Principal Component Analysis
GP	Genetic Programming
SOM	Self-Organization Map
ReLU	Rectified Linear Unit
GBDT	Gradient Boost Decision Tree
BP-NN	Back Propagation Neural Network
WOA	Whale optimization algorithm
LOF	Local Outlier Factor
CSNN	Cost Sensitive Neural Network
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative

List of Appendices

List of Tables

List of Figures

Chapter-1

Introduction

1.1 Fraud Detection Problem

Modern commerce is largely dependent on the use of e-commerce platforms and the various electronic transactions that are conducted through them. Online banking has become more prevalent due to its numerous advantages, such as lower fees, better customer service, and faster processing times. Security is also a major concern for customers when it comes to online banking. Due to the rise of fraudulent transactions, many customers are afraid about their financial security. To prevent these types of transactions, banks should develop fraud detection systems that can detect suspicious transactions.

An online transaction fraud detection system classifies the transactions into two categories: fraudulent and non-fraudulent. The systems use a comparison method to verify the transactions. An efficient fraud detection system can detect high-risk transactions and prevent them from happening. Rule-based systems are also used to prevent fraud. Rule-based systems use predefined rules to identify patterns of fraudulent transactions.

The rise of online banking has become more prevalent due to its numerous advantages, such as lower fees and faster processing times. It eliminates the need to go to a physical bank for every transaction [1]. Money transfers can be made from anywhere using a mobile phone or a computer. The number of people who can perform cashless transactions has increased due to the implementation of the UPI system. [2]. The ability to perform money transfers from other accounts without having to go to a physical bank has also increased the number of people who are willing to use UPI.

Cashless transactions are becoming the norm in today's world, especially for small businesses and enterprises. However, the number of cases of internet fraud has also increased [3].

The increasing number of attacks against banks and their customers has resulted in the loss of money for both the institution and the customers. As hackers develop new techniques to infiltrate a network, they usually wait for the transactions to occur before they expose themselves. The security measures taken by banks and their customers are regularly updated, but the hackers are still able to gain unauthorized access to the network. They then use this opportunity to steal data and carry out fraudulent transactions. The banks and their customers are not aware of the details of the attacks until they receive a report from the affected customer [4].

To prevent fraud, banks implement various security protocols designed to prevent unauthorized users from accessing their accounts [5]. Unfortunately, these security measures can sometimes fail due to the nature of the attacks. In most cases, they are still vulnerable to exploitation since the fraudsters can still gain unauthorized access.

The most common type of fraud that takes place is phishing, where the account details of the users are stolen including their authentication details. Phishing takes place in form of false emails or websites that mimic the original bank communication. The links in the mail redirect the users to an illegitimate site that is designed to look like the original website. When the users enter the authentication details on the website, these details are transferred to the fraudsters who use this to gain entry into their account [6].

As per RBI's annual report [7] published on 27 March 2021, online frauds (Card/Internet) have been increased during the past financial years and the amount involved in these frauds is also huge. The card/internet frauds of Rs 1 lakh and above reported during financial year 2018-19 were 1866 which is 27.5 % share of total frauds reported in all areas of operations like advances, off-balance sheet, foreign exchange transactions, card/internet, deposits, inter-branch accounts, cash,

cheques/demand draft, etc., clearing accounts, etc. and others. The total loss of amounts involved due to these 1866 frauds is 71 crores. In financial year 2019-20, card/internet frauds reported are 2677 which is 30.08% share of total frauds reported in all areas of operations and total loss of amount is 129 crores. In 2020-21, count of these frauds is 2545 which is 34.6 % of total frauds and total loss of amount is 119 crores. Thus, the share of these online frauds i.e., credit/internet among all types of financial frauds has increased in the last financial year as compared to previous financial years. The same has been shown in Figures 1.1 and 1.2.

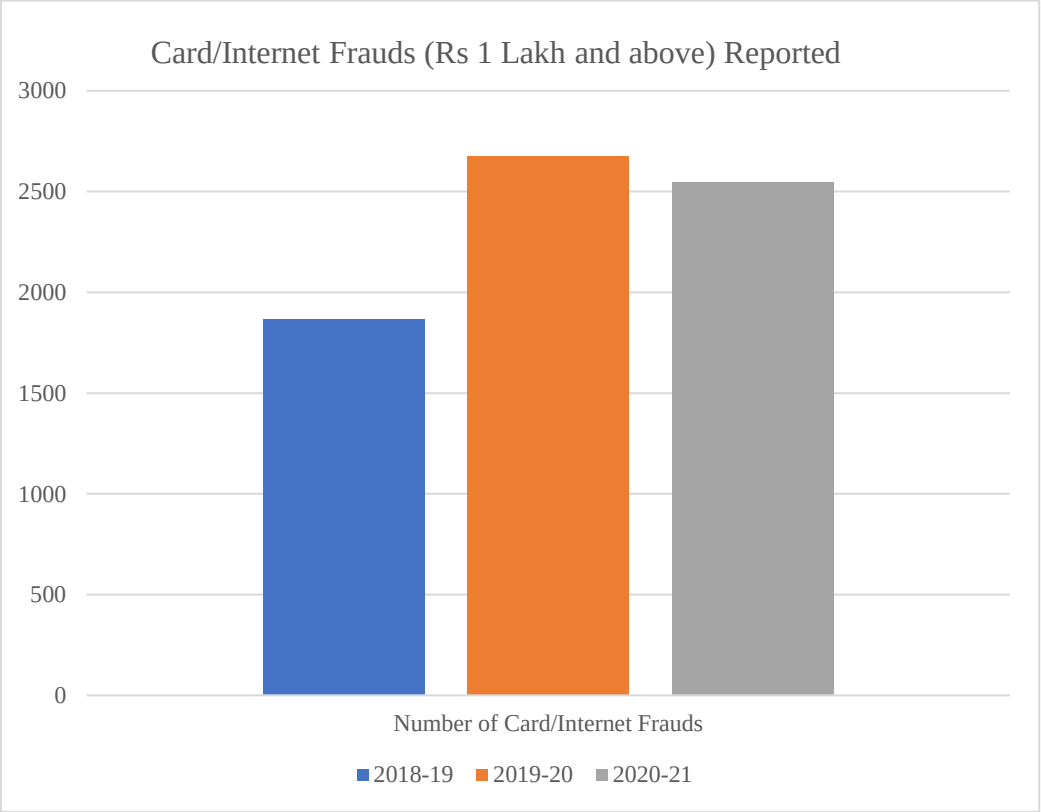

Figure 1.1: Card/Internet Fraud Cases reported in India as per RBI report [7]

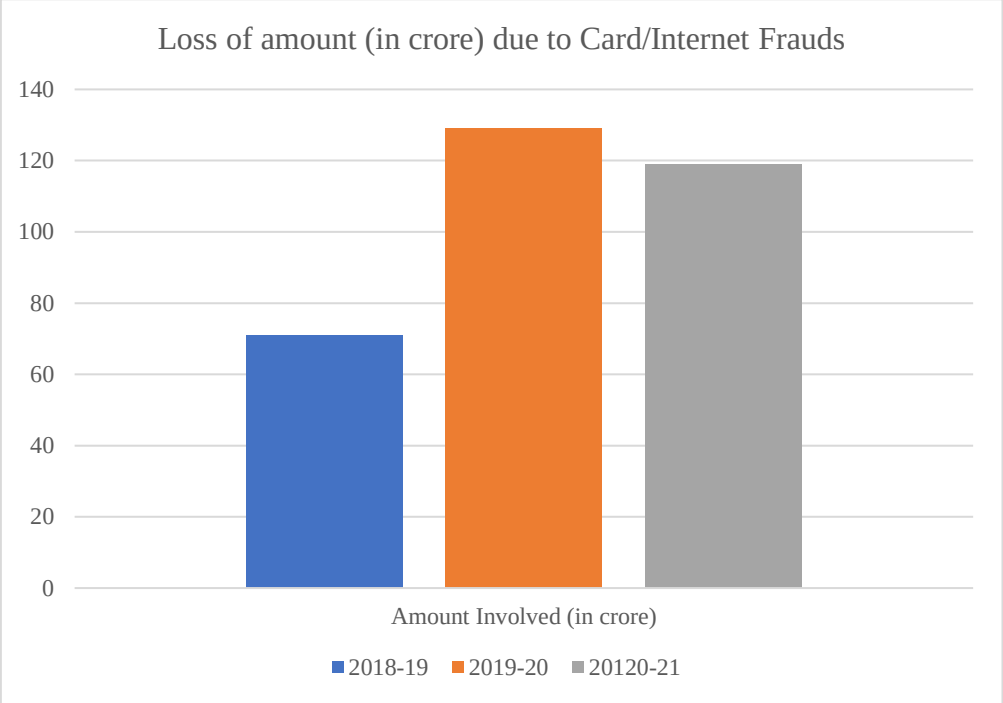

Figure 1.2: Loss of amount due to Card/Internet Frauds as per RBI report [7]

Due to increasing number of customers demanding more security measures for online transactions. There are various techniques available in market to protect their online transactions. Various techniques available for detecting and preventing credit card fraud. Some of these include HMM, Data mining, Biometrics, and Neural Network. However, some of these methods are not always effective at detecting and preventing transactions. Due to the increasing digital transactions, the central bank has started to enhance the security measures against cyber risk. This is in order to prevent the exploitation of cyberspace. With the help of e-commerce, people can shop for various products and services online. It is very beneficial for today's generation. One of the main advantages of digital market is the ability to provide unlimited Internet access to rural areas. Also, it is very affordable for consumers. This is the reason why rural areas are becoming more digital savvy. Due to the huge potential of India's market, digitalization has gained acceptance from international companies.

This is the reason why India is considered as a centre of excellence for e-commerce. Due to the increasing use of credit cards, fraud has become a major issue in the card business. It is difficult to provide a comprehensive loss estimate due to the reluctance of banks and companies to disclose the amount of losses caused by frauds.

An accurate and reliable method for detecting credit card fraud involves analyzing the data collected by a merchant. This process is carried out by analyzing the various attributes of a transaction (e.g., card identifier, date and amount of transactions, etc.). The Expert Driven approach uses rules formulated by fraud investigators to predict the likelihood that a new transaction will be fraudulent. The advantages of expert rules are many as they are very easy to develop and understand, they explain the reason for an alert, and they exploit domain expertise. However, they have some drawbacks: they are subjective and cannot predict all types of frauds, and they require human monitoring and supervision. A different way to detect fraud is by means of Data Driven methods. This method involves setting up a machine learning system that learns from the data in unsupervised or supervised mode. It then learns the most probable fraudulent patterns. Data Driven approaches can also benefit from learning complex fraudulent configurations and predicting new types of fraud. However, they can also cause errors and limit the number of fraudulent activities.

An efficient system that can detect fraud in real time are very critical to banks' operations. The system should be able to identify fraud in real time by matching the historical transaction data and prevent such fraudulent transactions in future as well. The characteristics of past transactions should also match those of the current transactions in order to make them safe.

To compare these transactions, artificial intelligence is required to get trained and classify the transactions correctly. An artificial neural network (ANN) is structured similar to the human brain. The layers are connected like the actual human neurons and can learn and store the trained data in form of models. These models act as the brain and hence will contain the trained memory which will be accessed during the testing phase. The neurons contain weighted values which are multiplied by the

values to give a weighted function. The weights are multiplied based on the distance travelled [8].

1.2 Types of Online Frauds

Online frauds are of distinct dimensions, forms, and intents. Some common types of online fraud include:

- **Advance payment fraud**

In advance payment fraud, fraudsters convince victims to make payment for receiving something that is valuable, however, they do not provide anything that is valuable to the victim. In this type, fraudsters collect only money from victims and deceive them without providing the requested service [9].

- **Phishing**

Phishing occurs when fraudsters utilize email or fraudulent SMS to lure users to bogus websites, wherein users are asked to provide personal, financial, and confidential data like account numbers, passwords, or transaction details [10]. For phishing, the emails or SMS are sent by fraudsters on behalf of e-shops, credit card industries or banks, requesting users to update or change their profile/account defaults. The phishing SMS/email texts are persuasive, instigating users to believe in their origin. In these texts, the hyperlinks, and logos of companies, mirroring the original ones, are used to enhance user trust. When the user proceeds through such websites and enters the requested personal details in those links, the fraudsters smartly take advantage and utilize those details for their vicious needs.

- **Online auction fraud**

Online auction (OA) fraud is an important variety of fraud. This sort of fraud mostly affects clients/users during the bidding activity performed online for products and goods. In this fraud, fraudulent transactions occur within the framework of an online auction website. In other words, this fraud is generally committed through different auction sites. Such auction sites bring about a hundred million goods together for trade, with multiple buyers and sellers. The trade generally includes sales of items like

vehicles, clothes, books, consumer electronics, antiques, etc. Certain sellers often fail to deliver the items paid for by the purchasers or sometimes deliver items other than those displayed on the sites. These sites, moreover, create convenient terms for committing online fraud. In many instances, auction websites are employed as platforms specifically for selling stolen items [11]. Also, they are utilized to withhold items or sell items which do not exist. In OA fraud, bidders taking part in auction trade are informed customarily that they have won the auction and are made to pay the amount, however, bidders, after payment, do not gain the goods paid for. It is hard to identify fraudsters involved in this sort of fraud, because they typically register on auction sites under counterfeit identities and confuse law enforcement personals and general clients [12]. Another common OA fraud is selling goods which misinterpret their authenticity or value [13]. The misinterpretations generally can include boasting retail values (in order to characterize the auction costs as significant savings), misrepresenting an item's condition as brand-new, selling counterfeit items or advertising goods in inferior working condition as completely functional. The OA fraud may also include shill bidding i.e., simulating the good's price on an auction through placing bogus bids [14]. The shill bidding is executed by utilizing several counterfeit identities or colluding with another trickster to participate on several bidding simultaneously.

- **Online investment theft/fraud**

In online investment theft, tricksters often persuade online customers to invest certain amount in various non-existent industries situated abroad. Alternatively, tricksters motivate online customers to buy shares in such (non-existent) firms. Fraudsters, in this sort of fraud use bulletin boards, investment newsletters, chat rooms and mass emails for attracting clients. They adopt a method called 'pump-&-dump', wherein an individual contacts another individual claiming to contain 'inside information' regarding a stock exchange listed organization. This makes that individual to purchase stocks, because he/she anticipates a rapid and good income. With the fresh high price, the fraudster 'dumps' his/her stock in company so as to cash in on short-term rise. When the stock price decreases, the trickster acquires profit at the cost of cheated clients.

- **Identity fraud**

Identity fraud or identity theft occurs when a trickster acquires and exploits personally recognizable and financial data of another individual. Identity theft is a form of illicit impersonation. Another example of identity fraud includes account takeover, wherein a fraudster performs fraudulent activity through acquiring access to a customer's account by borrowing and hacking information like security codes and passwords. In account takeover fraud, the trickster may even tamper email addresses of victims and details linked with users' account without the real owner's knowledge. The other sort of identity theft/fraud is synthetic theft. In synthetic fraud, artificial details and true data are merged to create a fresh personality. The chief purpose of synthetic fraud involves making illicit transactions and establishing bogus accounts.

Identity fraud is dangerous, as the consequences (may take many years to amend) involve rejection of user for credit owing to bad reputation, closing of bank account, receiving bills mainly for services that user/victim never used, and blame put on the victim for debt which victim did not incur. Identify fraud, unlike robbery, can go entirely unrecognized before the real user/victim experiences an explosive loss. The warning signs for realizing that one has become a victim include rising debt on one's credit card or unknown transactions.

- **Card testing**

This fraud occurs when someone acquires access to stolen card numbers via receiving card information from malicious web and through theft. Despite having card numbers, they are unaware of a) whether transaction can be successfully completed using card numbers, or b) whether it has any limitations. Thus, for testing, fraudsters initially make some minor test purchases and examine whether card numbers can be employed to make complete transactions. After they realize that a card works, fraudsters will start making expensive purchases. Finally, the initial minor purchase testing strategy often goes unidentified. Victims realize that they have been cheated after fraudsters make large purchases.

- **Clean fraud and chargeback fraud**

Clean fraud is executed with the credit card that is filched from an authentic customer and is employed to make e-purchase. Here, the card holder's details and filched card is employed to perform the fraud which appears like a licit purchase done by a valid customer. In chargeback fraud, also called friendly fraud, the user keeps the products/goods purchased online, but still requests for a repay stating payment being done twice or purchase never done or item never received. Chargeback frauds often occur owing to hacked payment data and filched credit cards. Fraudsters employ this data for performing fraudulent actions or purchases and even shipping goods to their address or to the tampered address.

- **Triangulation fraud**

In triangulation fraud, the trickster establishes a bogus online shop providing products/goods at low prices. This sort of fraud involves three participants namely bogus online store, stolen data and unsuspecting user. Here, once the user buys the items, the bogus merchant immediately sneaks user's card details. The chief motive of creating web shops here is to gather credit card information of users making purchases or visiting these sites. The fraudster after acquiring card details, cancels the received payment on user's card.

- **Formjacking**

In formjacking, the fraudster learns the operation of transaction site's security system and injects software into javascript which can intercept user's card details when making an e-purchase or online purchase. Formjacking works mainly for poorly constructed sites having code vulnerabilities.

- **Payment fraud**

Generally, payment fraud involves counterfeit cards, stolen cards and lost/misplaced credit cards. In payment sort of fraud, the tricksters' complete the payments whereas card owners have to pay these bills. Payment frauds mainly occur on vulnerable websites and are chiefly employed in transactions not requiring physical presence of cards.

- **Interception fraud**

In interception fraud, after placing orders, the package is intercepted by fraudsters and goods are taken by fraudsters for themselves. In this sort of fraud, fraudsters may ask the company's client service representative to change the package's (order) address before shipping that package. Here, fraudsters aim to obtain the goods/package while the package payment is already done by the victim. Sometimes, fraudsters may even contact the courier (shipper) to reroute the goods to their address.

1.3 Reasons for Cause of Online Frauds

The chief reasons for online fraud occurrence include fairly easy techniques for trickers, hackers or criminals to filch the required information, easy modes for purchases of hacked data on black market for fraudsters, inadequate law enforcement or punishment for this sort of misdeed. There are numerous reasons which lead to frauds. Some of them include:

- **Advent of fresh marketplace platforms**

Growth of digital channels, social media, alternative transportation, vacation rentals and food delivery apps have reformed almost every domain. Throughout this year, nation-wide quarantines have led to a greater surge in online application usage, with users ordering the products/goods delivery. With the expanding number of digital services, marketplace platforms and their growing popularity, particularly in recent days, fraudsters have switched their plans to take best advantage of augmenting online marketplace and in-app purchases.

- **Increasing online payments**

Along with performing more transactions in e-marketplace platforms, consumers are also employing eWallet and peer-to-peer (P2P) payment applications more often. As these online payments aid in faster transactions, consumers often use them to perform their simple to complex transactions. However, as the majority of P2P transactions occur between an anonymous entity and consumers, the fraud occurrence is high.

- **New user expectations**

Present-day users expect secure transactions. But, they relinquish any transaction which consumes longer time, needs too much information or is highly sophisticated. Indeed, 92% of users expect a rapid, congenial experience along with receiving one which is as secure and trustworthy as possible. These deep expectations are making retailers and banks to juggle avoiding losses with retaining fraud prevention actions from repudiating good users and transactions. Fraudsters understand the difficulties these corporations face and smartly take advantage of those which fail to establish a proper balance of safe, yet frictionless user experiences.

- **Increasing online banking services**

Nowadays, users demand more mobile and online services from financial institutions. Hence, legacy banks have now switched to digital transactions. Banks are now performing more transaction approvals and account onboarding online and reducing in-person transactions to provide digital-native and simple-to-use experiences. However, fraudsters make misuse of these growing mobile banking services for their vicious desires and deceive common users.

- **Advanced software**

Professional hackers employ improved anti-piracy software, which prevents common users or customary browsers from identifying them. These hackers also create virtual machines and Ips for easily hacking user details or committing fraud.

- **Technological advancements**

Currently, fraud has accelerated and has grown more complex owing to the augmentation of e-commerce, computing power and mobile payments. Majority of the similar technologies that industries rely on to advertise and rapidly launch new services and goods are also being employed by fraudsters. Fraudsters can more smoothly commit fraud adopting on demand, cheap compute power and through manipulating fraud identification systems.

- **More complex fraud tactics**

Owing to the growing amount of information breaches in recent years, tricksters can

more smoothly access personally identifiable data (PID) and utilize it against the users. For example, fraudsters combine bogus and real data to establish synthetic, new identities, which are tedious to recognize. Further, they set up transaction cards and open bank accounts, acting like authorized users. After establishing credit scores, tricksters ask for larger loans and huge credit limits and simply discontinue paying. Fraudsters/tricksters also leverage PID for user account takeover. Through using credentials and passwords acquired via social engineering or data breaches, they acquire control over user accounts and further misuse them or make fraudulent mobile purchases. These fraudulent transactions can include minor transactions like purchasing groceries using the victim's debit card or major ones like exploiting the victim's account to buy loans.

1.4 Preventive Measures for Online Frauds

Protection of online transactions from fraudulent actions is possible through recognizing various fraudulent activities. Fraud preventive methods can lower the fraud threat and assure that fraud does not affect the business. Some significant preventive measures against fraud occurrence include:

- **Conducting regular site security checks**

Users should regularly conduct site security checks to find flaws, if any, in their security configuration/ framework before fraudsters target and identify them. These security audits conducted often can aid in a) identifying whether communication or transaction during e-purchase activity is properly encrypted, b) confirming whether the secret codes employed for File Transfer Protocol (FTP) access, database and admin accounts are robust enough, c) confirming whether purchase making website/application is security standard compliant and d) confirming whether the site or transaction platform is being regularly scanned for malware.

- **Using address validation service**

Credit card issuing banks typically render an address validation service for detecting dubious real-time card transactions and preventing card frauds. An address validation service audits whether the transaction card user's billing address matches the billing

address details of cardholder. This validation is executed for authorizing the card transaction. In case of address mismatch, system either flags the transaction or declines it and performs further investigation. Thus, this service can prevent fraudulent transactions.

- **Utilization of device identification software**

Generally, device identification software aids in identifying and tracing the devices which request a transaction. The software of this type is valuable for ascertaining any transaction's authenticity and can trace the transaction's source in case of doubt of fraudulent activities.

- **Use of smart geo-location**

The smart geo-location technology assists in tracing and tracking user's location during transactions. The fraudulent action tracing through geo-location can easily identify individuals at any place/location when they are making any smart transaction. The smart geo-location technology is also productive when a trickster tries to hide location or identity.

- **Using Hypertext Transfer Protocol Secure (HTTPS)**

The HTTPS is basically a secure HTTP version, which exchanges information between the user's web browser and an e-purchase store. It encrypts information for protecting sensitive information like usernames, card numbers and addresses, thereby securing confidential user details from fraudsters. Utilization of HTTPS prevents fraudsters, hackers, and criminals from easily viewing the user's transaction details.

- **Utilization of digital signatures**

Digital signatures are mainly considered as the digital tantamount of conventional handwritten signatures [15]. Adoption of digital signatures for many online transactions are helpful for avoiding frauds, this is chiefly because they are tedious to forge. As these signatures employ cryptographic methods, fraudsters cannot easily forge them.

- **Use of email authentication**

Email authentication is another significant means for preventing online fraud. This approach determines user's details through connecting emails with information, mainly demographic details provided by the users. Thus, verification of these details through email authentication and validation of actual owner identity can assist in preventing transaction frauds.

- **Use of reverse lookup mechanism**

Reverse lookup mechanism mainly relies on public data (preserved in government records) authenticity. This mechanism involves cross-checking of phone data and address given by user via third-party sources in public documents/records. This mechanism thus verifies potential user's details with user's records in government database or public file. Its efficacy is chiefly dependent on the presumption that public databases are unfeasible to be tampered or misrepresented.

- **Utilizing anti-fraud solutions**

Today, various software solutions for fraud prevention are available. These tools differ broadly based on work involved in ongoing management and installation. The major anti-fraud solution categories include:

- Preliminary tools

Preliminary tools utilize machine learning schemes for recognizing fraudulent transactions via IP geo-location. These tools also verify addresses, perform device fingerprinting and audit email addresses for detecting fraudulent actions. They perform single functions and very specific ones at a moment.

- Mid-range tools

Mid-range tools provide a broad diversity of functions like auto dwindling of high-threat orders, chargeback guarantees and protection against account takeover and fresh account fraud.

- Top-range tools

Top-range tools provide almost everything which other tools provide along with automatic decisions, policy abuse prevention, manual scrutiny of dubious transactions and outsourced event management. These tools ensure that no legitimate order is mistakenly declined by software.

- **Card security methods**

Checking the verification number (which is unique to each card) can ensure the card's identity and cardholder's identity. This can ratify that the individual claiming to be the card's owner is the real owner and is having the transaction card [16]. Along with the card's verification number, cardholder's other details like address, location are also utilized. This can prevent transaction card frauds executed online, particularly when fraudsters have somehow extracted details using the user's history.

- **Monitoring transaction site for dubious activity**

Monitoring transaction sites for dubious activities greatly aids in safeguarding users from fraudulent actions. Monitoring transactions and accounts for warning signs like inconsistent billing, inconsistent user's physical location and shipping data can assist in preventing major frauds. Moreover, constant site monitoring aids in guarding businesses from potential dubious activity which escalate into quantifiable dicey frauds in future.

1.5 Fraud Detection Parameters

One of the biggest challenges facing financial institutions is how to prevent online transaction fraud. The losses caused by the fraudulent activities are quite negative for consumers and merchants. The legislation that is implemented to prevent this type of activity is also beneficial for the institution. Different techniques are also analyzed during an online transaction to detect fraud. The various parameters [17] like Transaction Amount, Card Address, Location of customer doing transaction, IP address, blocked list of users etc. that are used by the financial institution to identify the most likely fraudulent activities. These parameters are matched with the previous spending patterns derived from transactional history of the user and also with the supporting data being regularly updated by the financial institutions to prevent frauds

like suspicious/malicious user's list, blocked IP addresses etc.

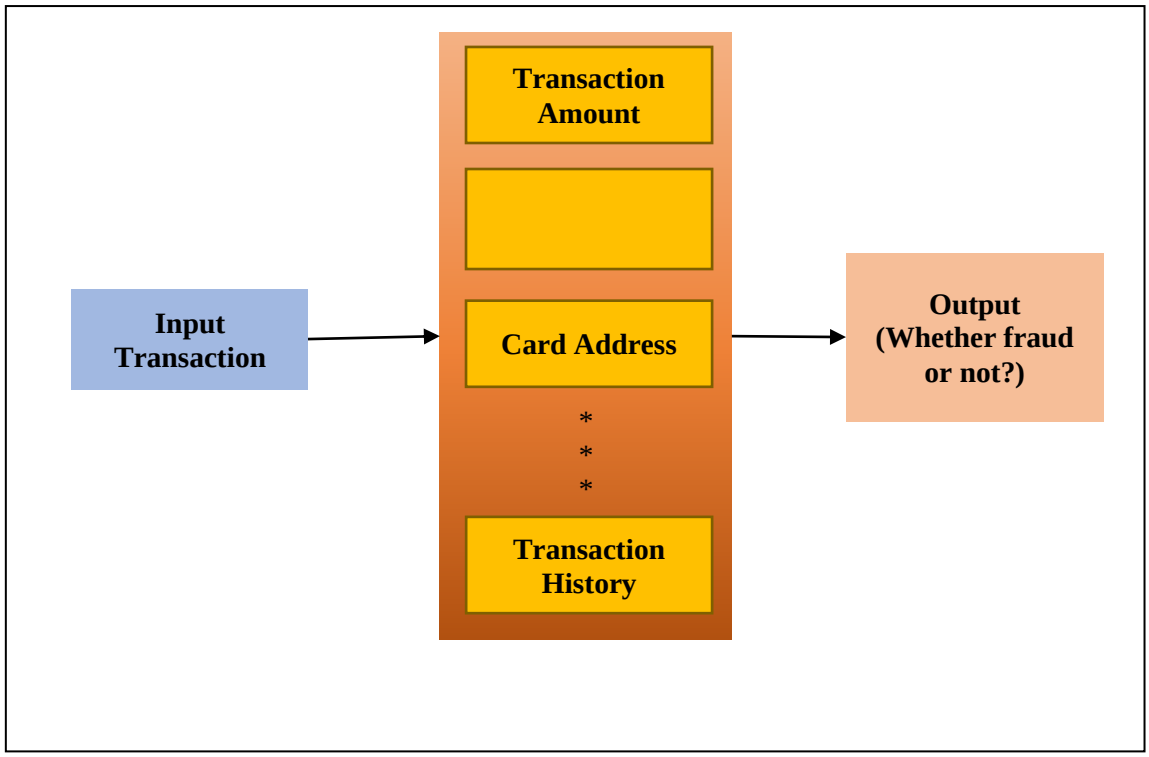

Figure 1.3: Parameters checking for Fraud Detection

The above Figure 1.3 displays that when a transaction occurs or being attempted, the various set of parameters are checked by the system to detect whether its fraud or not. There can be many numbers of parameters being checked by the system.

1.6 Deep Learning in Fraud Detection

With the emergence of fresh and innovative technologies in the present world, the menace of online scam or malicious activities is increasing. In every facet of advancement, there exists a continuous threat of getting duped in one way or the other. The reasons for a contender to threaten the institution/organization or an individual can be different. Some of them include personal animosity, demeaning reputation, defamation, and monetary gain. These incidents relate to fraud. Importantly, fraud is any felonious activity executed by an entity which causes loss to any organization or an individual [18]. It can occur in diverse fields and in several

ways. Lots of online transactions occur everyday. Transactions of this type encompass monetary exchange performed online. These transactions are prone to several sorts of frauds. Frauds occurring during diverse online transactions (online frauds) generally exploit online facilities for performing dubious transactions with the motive of defrauding or deceiving persons, organizations or governments [19]. An efficient means to handle fraudsters and fraud is to examine and detect frauds. An activity which involves identification and detection of frauds which cause loss to target entity (an organization or a person) is known as fraud detection [20]. Recently, many technologies are available for identifying frauds. Deep Learning (DL) is one among such technologies which offers outstanding potentials for fraud identification and categorization [21-25]. DL is mainly a broader category of Machine Learning (ML) techniques that teach machines to perform sophisticated operations without any explicit programming. Among various DL schemes, Deep Neural Network (DNN) is commonly employed for fraud recognition problems.

The DNN comprises one or many hidden layers with computational nodes known as hidden nodes. For a DNN topology with a single output layer and two hidden layers, initially the input data enters the primary hidden layer [26]. The primary hidden layer's output is fed as input to second hidden layer. For multi-layered networks, the second hidden layer's output is fed as input to the succeeding layer and this process is continued for all layers. The final layer's output is then considered for evaluation. Every layer obtains outputs from past layer as inputs; therefore, the input signal moves forward on layer-by-layer base till the output layer. For fraud identification, firstly input information is fed to DNN layers, desired attributes, or parameters pertinent to fraud activity detection are extracted and then DNNs are trained well for predicting and categorizing non-fraudulent transactions and fraudulent transactions. There can be many hidden layers in DNN where each hidden layer of the network extracts meaningful information which then forward to next hidden layer as input (Figure 1.4). In this way, Deep Neural Networks (DNN) learn from the data they collect and improve their performance over time.

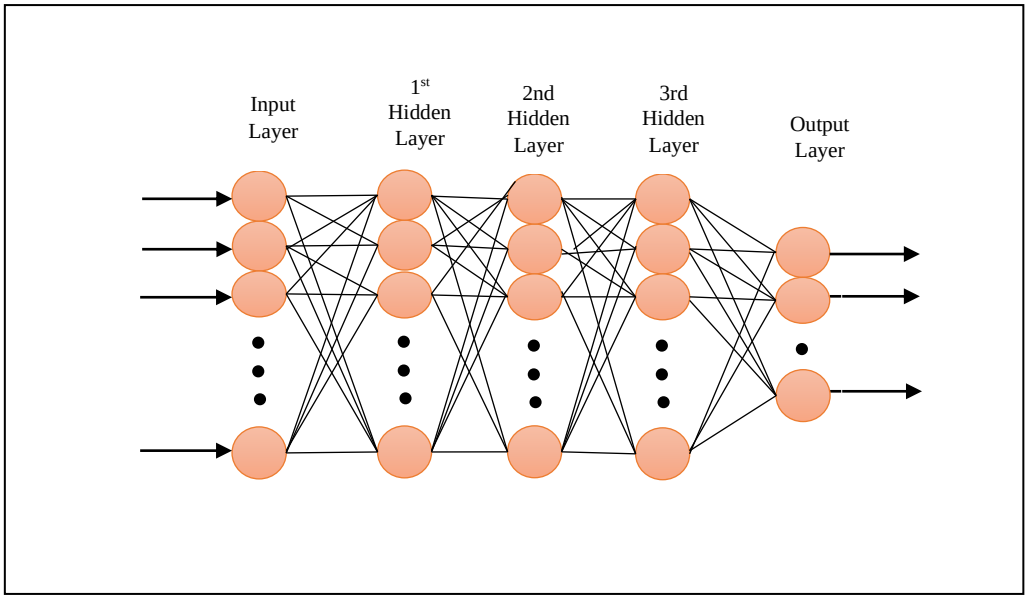

Figure 1.4: Layers of Deep Neural Network (DNN)

1.7 Machine Learning

Machine Learning (ML) is a subcategory of Artificial Intelligence that can smartly determine patterns in information and further exploit those patterns for future consequences or making decisions through considering certain conditions. In ML, the computer constructs training information through the provided data, which helps with decisions and predictions. ML schemes for fraud identification can be segregated into unsupervised and supervised models. Unsupervised schemes search correlations and patterns in raw information and perform prediction without any additional labeling while supervised schemes demand a great deal of data like non-fraudulent and fraudulent transactions in equivalent amounts for model training. Fraud detection with ML becomes feasible owing to ML algorithm's potential of learning from historical or past fraud patterns and recognizing them in upcoming/future transactions [27]. They appear more productive compared to humans with reference to information processing velocity. Also, they are capable of finding complex fraud traits which humans simply cannot identify. Fraud identification using ML begins with collecting and partitioning the information. Further, involving training of ML algorithm or model for predicting fraud probability. The ML scheme should be eligible of discriminating between good

information (comprising genuine users) and bad information (comprising fraudsters). Segmentation of this information helps the ML model to perceive better and render results efficiently. Further, feature extraction is carried out. The features aid in determining signals which aid in recognizing frauds. For fraudulent action discoveries, the substantial features include users' past transaction details, preferred payment modes of users, users' location for transactions, users' identity (card numbers, email addresses, etc.) and users' network details (payment details and phone numbers entered/registered with the account). For discriminating between the normal transaction and fraudulent transaction, ML method is trained via learning dataset, to make right predictions. Training set aids the ML model in comprehending and understanding the defined algorithm. After completing machine training, the exact framework needed for identifying frauds is obtained.

- **K-Nearest Neighbor (KNN)**

KNN is a supervised learning scheme where the outcome/result of a fresh example query is categorized depending on majority of KNN category. Generally, KNN method's performance is largely influenced by a) distance metric employed for locating the closest neighbors, b) number of neighbors employed for categorizing fresh sample and c) distance rule adopted for deriving a categorization from KNN [28]. In KNN, distance between two information instances is quantified in diverse ways i.e., Euclidean distance (for consistent attributes) and a matching coefficient (for categorical/absolute attributes). For fraud identification using KNN, any arriving transaction is categorized through estimating the closest point to fresh incoming transaction. The transaction here is considered fraudulent, if the closest neighbor is recognized to be fraudulent.

- **Decision Tree (DT)**

DT is an evolved ML algorithm category employed to automate mainly establishment of rules for specifically classification jobs [29]. DT approaches can be adopted for predictive modeling or categorization problems. They possess comprehensibility, highly preferential feature choosing ability, data categorization without involving much tedious calculations, both discrete and continuous data handling ability and

dealing with incomplete or noisy data. DTs are principally a bunch of rules that are chiefly trained through examples of various frauds encountered by clients for fraud identification. Construction of a DT neglects irrelevant attributes and requires no extensive data normalization. It gives the fraud's probability score depending on previous scenarios.

- **Logistic Regression (LR)**

The LR employs a cause-effect dependency to design structured datasets. LR, when applied to fraud identification, provides results through evaluating predictive power of combination of parameters/variables or individual variables as portion of a huge fraud strategy. The authentic transactions in LR are compared with fraudulent ones for creating the model. Then, this model predicts the status of a new transaction (i.e., whether it is authentic or fraudulent) [30].

- **Random Forest (RF)**

The random forest algorithm was first presented in 2001 by L. Breiman as a general-purpose regression method [31]. The random forest algorithm combines several random decision trees and combines their predictions by averaging. The algorithm can be easily adapted to large-scale problems and can return measures of variable importance. One of the main advantages of random forests is their ability to handle missing data.

- **Light Gradient Boosting Machine (LGBM)**

The Gradient Boosting Decision Tree (GBDT) is a widely used machine learning algorithm that has been implemented in various implementations. Light Gradient Boosting Machine (LGBM) is a novel GBDT algorithm that can handle large number of data instances [32]. The experimental studies have shown that LightGBM can outperform both Stochastic Gradient Boosting (SGB) and eXtreme Gradient Boosting (XGBoost) in terms of computational speed. It can also handle large number of features without sacrificing its memory consumption.

1.8 Deep Learning vs Machine Learning for Fraud Detection

There are many types of ANNs that are classified based on their working. When the number of neurons is increased into multiple layers, it is then known as a deep network. Deep learning contains networks that are capable of learning in an unsupervised manner from unstructured or unlabelled data [33]. They are also known as Deep Neural Network (DNN), and they can handle large data available in big data. This data will be very huge and takes a very long time for humans just to analyze the data. This can be easily comprehended by the machine.

A literature review on deep learning revealed that this type of learning system has better accuracy than machine learning. Due to its high computational capabilities, it can handle massive amounts of data. As we are aware that online payment history of a customer will contain millions of transactions, so the data utilized by the learning model for fraud detection will be big data [34].

Unfortunately, most machine learning algorithms can’t handle the amount of data that they’re required to handle. Performance of ordinary machine learning systems will typically plateau once they handle a certain amount of data. Deep learning systems can steadily increase as their data gets bigger. Their performance increases dramatically as more data is being utilized. This performance comparison of deep learning with other ordinary algorithms is shown in Figure 1.5. Thus, deep learning is able to learn and classify large numbers of transactions. Deep learning systems can handle large amounts of data without experiencing performance issues.

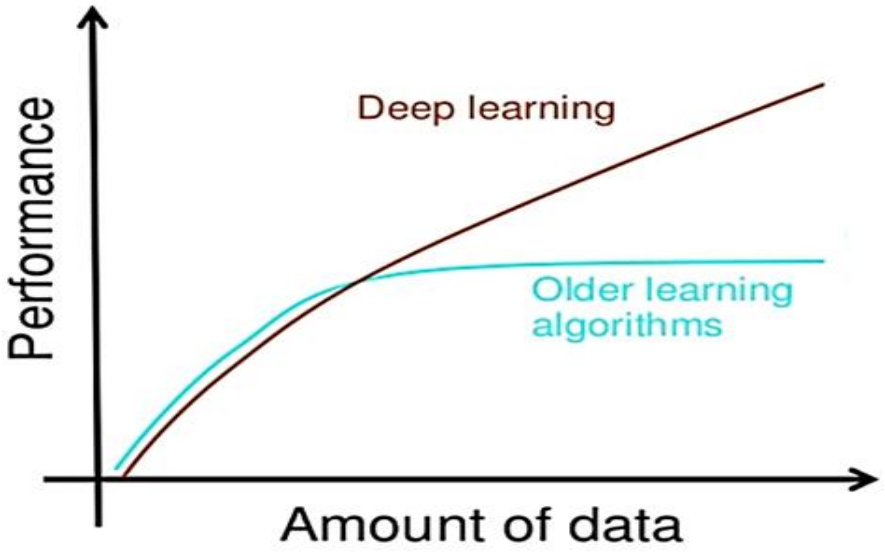

Figure 1.5: Comparison of Deep Learning with other learning algorithms [35]

A deep learning system can also learn and extract features from the raw data that it collects. This process is known as feature extraction. Hence, this work aims to detect fraudulence on online transactions using deep neural network (DNN). Due to the availability of data, deep learning has become popular in various domains such as computer vision and fraud detection. However, due to the secrecy of the data collected in online banking systems, research in this field has been limited. Due to the confidentiality of the data, the datasets available are typically limited and not available in their raw form due to transformation techniques applied on them before releasing for public access for research purpose. This makes feature extraction a major challenge for machine learning systems [36]. Due to the overlapping patterns of transactions, deep learning systems can easily transform and extract features without requiring additional features extraction techniques [37]. In feature extraction, input raw data is converted into a set of features [38]. Feature extraction techniques are mainly used to increase the accuracy of learning models. Selecting the appropriate feature extraction technique requires a lot of research and effort. Traditional machine learning models need hand-crafted features and hence they utilize feature extraction techniques to improve their performance whereas deep learning models have emerged as the automatic feature extractors along with the efficient classification models as they learn the features automatically from the input data. Hence, a lot of time and

effort is saved as they do not require hand-crafted features. However, the usage of deep learning models as feature extractors is understudied and needs to be explored more. Deep Learning Models are multi-layer perceptrons (MLPs) i.e., Deep Neural Networks (DNNs) which include a layered architecture of data in which high-level features are extracted from the last layers of the model and low level of features are extracted from the lower layers [39]. They use the backpropagation procedure while training [40]. The weights of the connections in the network are adjusted during the backpropagation process to minimize the difference between the actual output and desired output. The weight adjustments in the hidden units of internal layers lead to the representation of important features.

Machine learning models rely on the features of data on which they are trained to achieve a specific goal. Thus, they are dependent on feature extraction techniques to extract the features for them. On the other hand, the deep learning model is a multilayer neural network that can extract features from the raw data by itself and not much dependent on the feature extraction techniques to extract features for them [41]. Figure 1.6 shows the comparison of deep learning with machine learning.

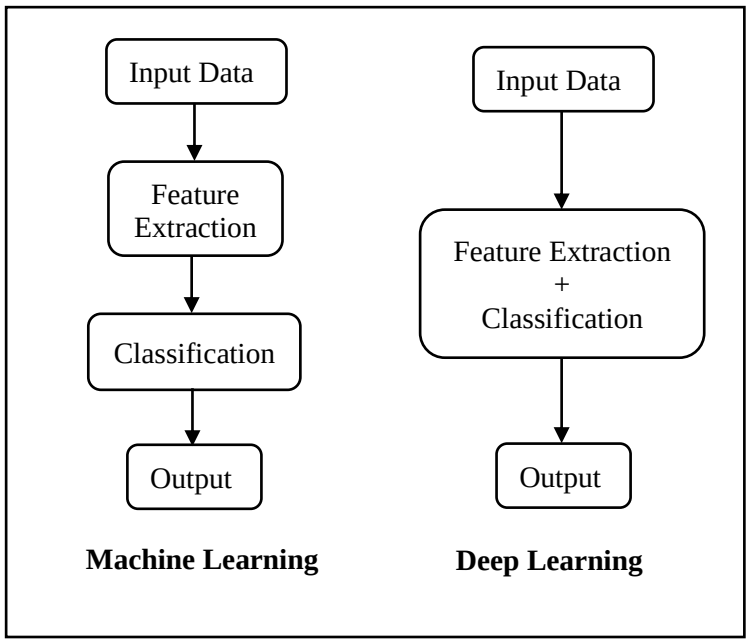

Figure 1.6: Machine Learning vs Deep Learning [41]

Hence, deep learning model can also be used in conjunction with other machine learning models to improve their performance [42]. There are various types of deep learning models that have been used in various domains as per their application. Deep Neural Networks (DNNs) have been used for fraud detection, anomaly detection, spam detection, natural language processing. Convolution Neural Networks (CNNs) have been used widely in computer vision and image processing. CNN extracts the features directly from images by utilizing two-dimensional convolution layers [43]. They have shown better performance in terms of feature extraction [44]. Autoencoders have also been used as feature extractors in many unsupervised tasks.

1.9 Problem of Class Imbalance in Fraud Detection

Class imbalance is a major problem when it comes to the transactional dataset. Class imbalance is usually not studied in deep learning systems to handle non-image data such as historical transactions. [45]. Many researchers have used data-level techniques that alter the dataset to handle the class imbalance method. The important information may get lost by modifying the data. Changing the learning model to handle the class imbalance can prevent the important information from getting lost which can be achieved using Algorithm – level techniques which need to be explored more [46].

Thresholding or threshold moving is rarely studied in deep learning with class imbalance [47]. The decision threshold should be adjusted in cases where the data is imbalanced to ensure that the prominence of minority classes is maintained.

Hence, an algorithm-based methodology by utilizing deep learning has been proposed for detection fraud transactions in data that is imbalanced in nature. The comparison of the same has also been done with the existing methodologies.

1.10 Deep Learning with Class Imbalance

All datasets are not ideal. Especially in fraud detection, where many transactions are occurring simultaneously, fraud transactions may be a few. So, to handle these real-world datasets which encounter a level of class imbalance, following methods are available.

1.10.1 Data level Methods

Sampling of the data is performed in these methods. They either reduce the frequency of majority class or increase the frequency of minority class to balance the dataset. Random undersampling (RUS) and random oversampling (ROS) are two most used data level methods in which we randomly select the samples from majority or minority class to decrease or increase their ratio to balance the dataset respectively.

Example: Suppose we have 1000 genuine instructions and 100 fraud transactions in the dataset i.e., out dataset is imbalanced in nature. Then, to balance it we perform either undersampling or oversampling.

In random undersampling (RUS), we randomly select 100 genuine transactions to make their count equal to the fraud transactions so that the frequency of both types of transactions is same i.e., 100 and total transactions become 200. On the other hand, in random oversampling (ROS), we randomly increase the count of fraudulent transactions so that the frequency of both types of transactions i.e., genuine and fraud is same i.e., 1000 and total transactions are 2000. Since, the data becomes balanced after performing RUS or ROS, hence default threshold (0.5) can be used to classify the transactions by the learning model.

1.10.2 Algorithm Level Methods

These methods don't modify the data as done in data-level methods. By changing the model's learning can also help improve the minority class's importance and hence these methods do the same. Algorithm-level methods that are used for making changes in learning include adaptive learning rate, output thresholding, and cost-

sensitive classification [48]. In this research study, we have assessed the various loss functions along with threshold moving.

1.10.2.1 Focal Loss (FL)

A new loss function [49] which handles the class imbalance problem by down weighting the easily classified class examples so that their contribution is low in the total loss value. Hence, easy examples correspondingly make less influence on the weight updates. Easily classified examples are those which hardly help to discriminate that class with other classes.

Cross entropy loss (CEL) is not suitable for acute class imbalance. FL modifies the cross-entropy loss (CEL) as shown in Figure 1.7 by down weighting easy examples. Thus, CEL has been multiplied with a modulating factor, $-\alpha_t (1-p_t)^{\gamma}$ to achieve it.

$$\mathbf{CEL}(p_t) = -\log(p_t) \tag{1}$$

$$\mathbf{FL}(p_t) = -\alpha_t (1-p_t)^{\gamma} \log(p_t) \tag{2}$$

$$\textbf{where, } p_t = \begin{cases} p, & y = 1 \\ (1-p), & otherwise \end{cases} \tag{3}$$

Here, the focusing hyper parameter $(\gamma \geq 0)$ is tuned to decrease the weightage of the easily classified examples whereas, α is a balancing hyperparameter which is generally taken into consideration to improve the importance of hard classified examples. The value of modulating factor becomes zero for easy examples whose probabilities approaches to one and hence reducing the impact of these easily classified examples on loss.

In this research work, we have performed experiments using different values of α, and γ = 2 has been used as it has shown superior performance [48]. The performance of

focal loss without balancing parameter α has also been checked.

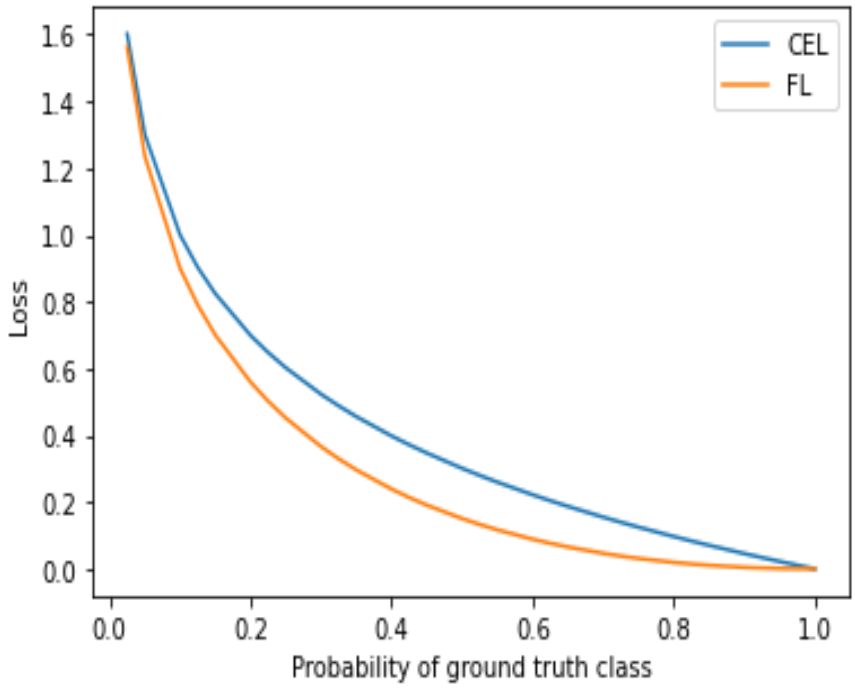

Figure 1.7: Comparison of CEL & FL

1.10.2.2 Weighted-Cross Entropy Loss (W-CEL)

X. Wang et al. [50] have proposed weighted-cross entropy loss (W-CEL) in their research work. They have balanced the dataset by applying a variable β to the loss function where β is given as:

$$\beta_P = \frac{|P| + |N|}{|P|} \quad \text{and} \quad \beta_N = \frac{|P| + |N|}{|N|} \tag{4}$$

β_P has been multiplied to positive and β_N is multiplied to negative class. Here, $|P|$ and $|N|$ are total number of labels '1' and '0' respectively in a batch. Thus, equation of W-CEL is obtained from equation (1) as following:

$$W - CEL(p_t) = -\beta \log(p_t) \tag{5}$$

1.10.2.3 Weighted-Focal Loss (W-FL)

R. Qin et al. [51] have proposed weighted-focal loss (W-FL) in which they have also utilized variable β to balance the focal loss (FL) as discussed in W-CEL. Thus, equation (2) has been modified for W-FL is as following:

$$\mathbf{W-FL}\left(\mathbf{p}_t\right) = -\beta\left(1-\mathbf{p}_t\right)^{\gamma}\log\left(\mathbf{p}_t\right) \tag{6}$$

1.10.2.4 Reduced Focal Loss (RFL)

Reduced focal loss (RFL) [52] is a novel loss function which was used for detecting objects in satellite images by the researchers and their solution got them first prize. They modified FL to decrease the importance of easy examples and alleviate the loss function's response towards hard examples. Easy examples are easy to classify by a learning model, hence they have probabilities less than a threshold value (say 0.5). Hard examples, on the other hand, are those with probabilities that are greater than a threshold value.

Thus, a flat weight value has been applied to hard examples to give more weightage to them in the loss similar as Focal Loss (FL) exponentially increases the weights of hard examples. The equation for RFL is given as following:

$$\mathbf{RFL}\left(\mathbf{p}_t\right) = -\mathbf{f}_r(\mathbf{p}_t,\ \mathbf{th})\ \log\left(\mathbf{p}_t\right) \tag{7}$$

$$\textbf{where,}\qquad \mathbf{f}_r(\mathbf{p}_t,\ \mathbf{th}) = \begin{cases} 1, & \mathbf{p}_t < th \\ \left(1-\mathbf{p}_t\right)^{\gamma}, & \mathbf{p}_t \geq th \end{cases} \tag{8}$$

Here, default value of decision threshold has been used i.e., ('**th=0.5'**). Only easy examples (probabilities for which are in the range of 0.5 to 1) has been multiplied with modulating factor $(\mathbf{1}-\mathbf{p}_t)^{\gamma}$ to decrease their weightage in the total loss value. On the other hand, no modulating factor multiplied to hard examples (probabilities for which are in the range of 0 to less than 0.5) i.e., they have been multiplied with a flat

weight value of 1.

Thus, RFL works as a dual function which switches to the FL for easy examples and then switches to CEL for hard examples as visualized in Figure 1.8.

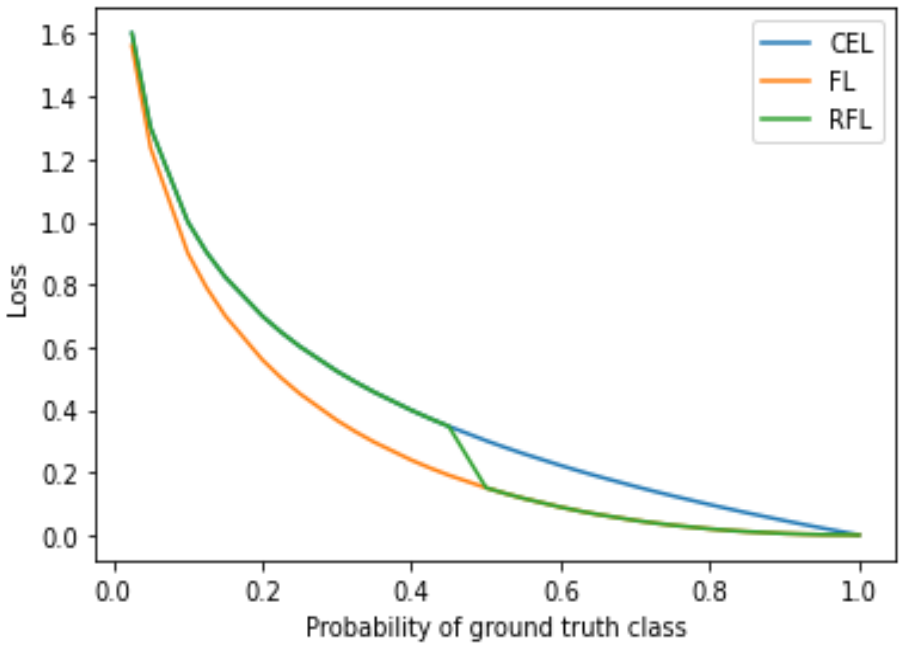

Figure 1.8: Comparison of CEL, FL, and RFL.

The weightage of hard and easy examples can be increased and decreased further in RFL as per the requirement by applying the weights to hard and easy examples separately as visualized in Figure 1.9 in which the weightage of hard examples has been increased and the weightage of easy examples has been decreased in the loss and hence named as Weighted RFL.

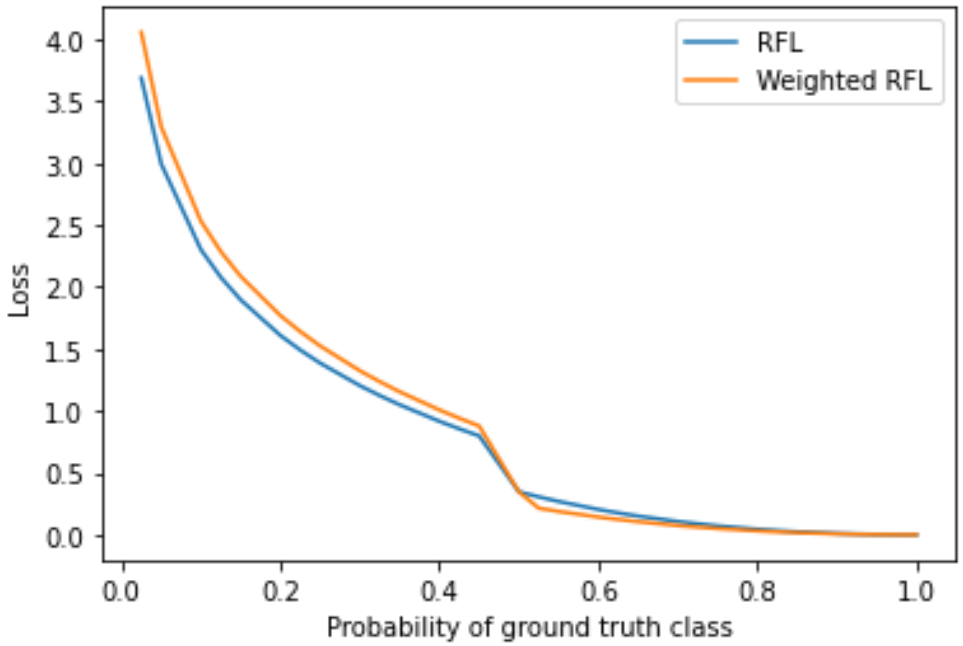

Figure 1.9: Comparison of RFL & Weighted RFL

The goal of our research work is to develop a system which can detect maximum number of frauds by increasing True Positive Rate (TPR). Thus, hard minority class examples having a probability range from 0 to 0.5 are required most attention in the loss as they are very critical. Thus, to maximize fraud detection rate, hard minority class examples should be assigned more weights than hard majority class examples.

The concept of the RFL is that instead of having all examples, hard ones have been assigned weights. But we modified the RFL as per our requirement. Hard-positive examples have been assigned more weights, while Hard- negative examples are assigned less weights. Hence, we renamed our modified RFL as **Weighted Hard Reduced Focal loss (WH-RFL)**. '**weight1'** multiplied to hard positive (minority class) examples and '**weight2'** multiplied to hard negative (majority class) examples and 'weight1' > 'weight2'. Easy examples (no matter positive or negative) have not been assigned any weight. For easy examples, WH-RFL will work as RFL. The equation of **WH-RFL** becomes:

$$\mathbf{WH-RFL}(\mathbf{p_t}) = -\mathbf{f_r}(\mathbf{p_t}, \mathbf{th})\ \log(\mathbf{p_t}) \tag{9}$$

$$\textbf{where, } \mathbf{f_r}(\mathbf{p_t}, \mathbf{th}) = \begin{cases} weight1 * 1, & \mathbf{p_t} < th \\ \left(1-\mathbf{p_t}\right)^{\gamma}, & \mathbf{p_t} \geq th \end{cases} \textbf{, for y = 1} \tag{10}$$

$$\text{and, } \mathbf{f_r}(\mathbf{p_t}, \mathbf{th}) = \begin{cases} weight2 * 1, & \mathbf{p_t} < th \\ \left(1-\mathbf{p_t}\right)^{\gamma}, & \mathbf{p_t} \geq th \end{cases} \textbf{, otherwise} \tag{11}$$

$$\text{where, } \mathbf{p_t} = \begin{cases} p, & y = 1 \\ (1-p), & otherwise \end{cases} \tag{12}$$

Various combinations of flat weight values were then tested to see how they would perform. The values ‘weight1’ = 2 and ‘weight2’=0.5 (‘weight1’ > ‘weight2’) has performed well by achieving maximum TPR and maintained the overall performance of the system.

The comparison of WH-RFL with RFL has been visualized in Figures 1.10 and 1.11 in which we have increased the weightage of hard positive examples and decreased the weightage of hard negative examples respectively.

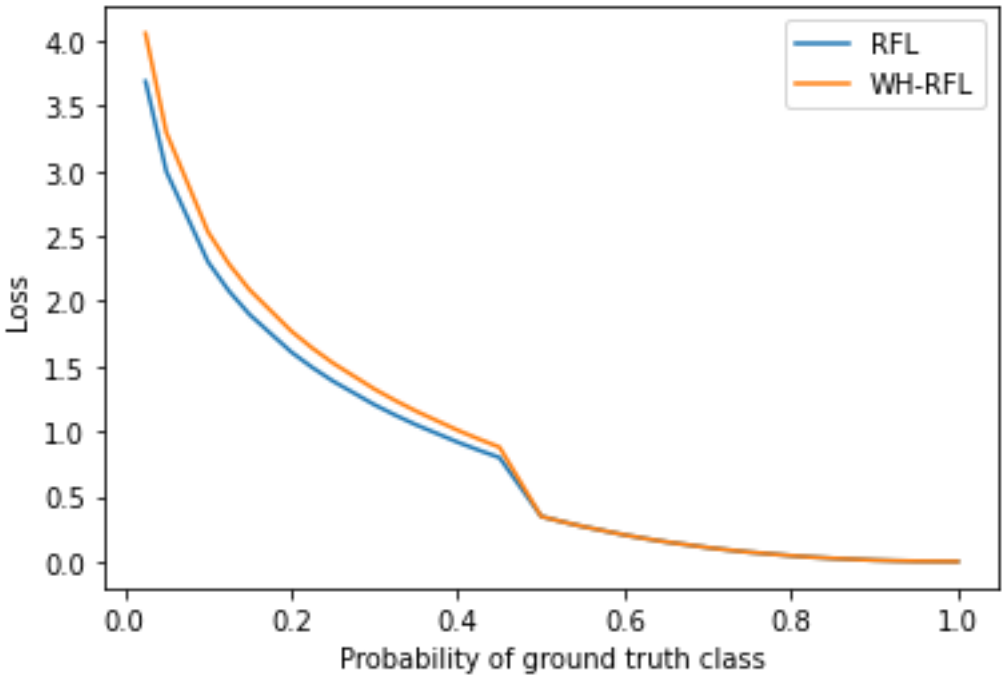

Figure 1.10: Comparison of RFL & WH-RFL for hard positive examples

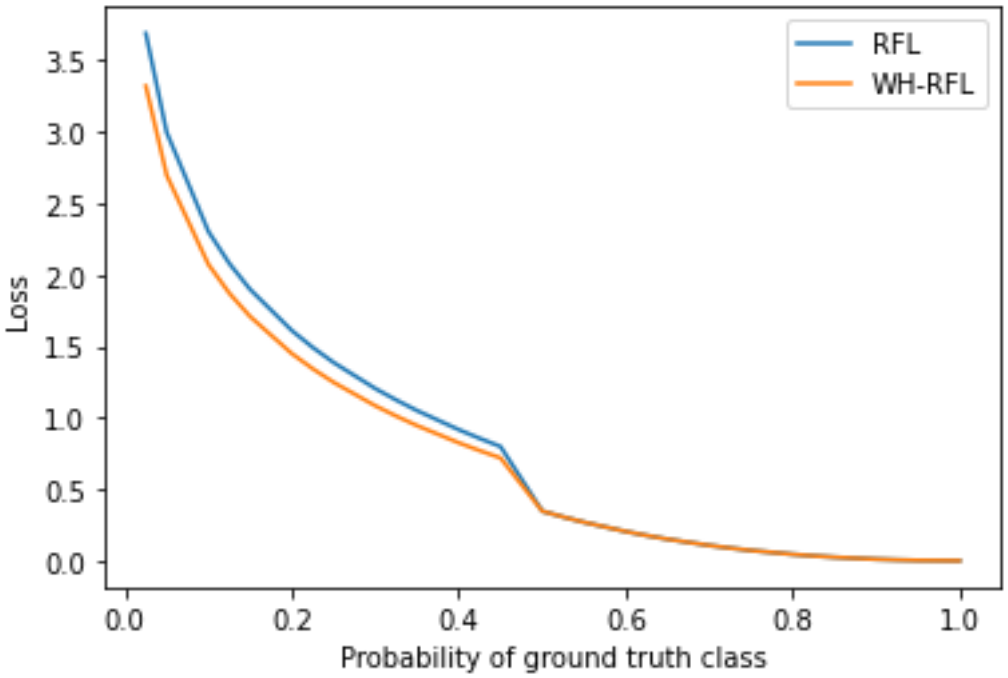

Figure 1.11: Comparison of RFL & WH-RFL for hard negative examples

How **WH-RFL** loss function is better in achieving high TPR i.e., can handle class imbalanced fraud detection data as compared to existing loss functions has been demonstrated with the help of following example.

Example: For instance, let's assume we have two types of transactions which are genuine and fraud. Genuine transactions are the majority class which are total three and fraud is the minority class which is only a single transaction. The estimated class probabilities (i.e., $1 - p_i$) of genuine transactions are respectively 0.2, 0.7, and 0.8. The estimated probability of fraud transaction is 0.2 (p_i) at epoch K.

For FL, RFL, and WH-RFL, $\boldsymbol{\gamma = 1}$ for easier explanation. Then, various loss functions values are calculated at current epoch (K) during training of the model:

$$\mathbf{CEL(K)} = -\mathbf{log(0.2)} - \mathbf{log}\,\mathbf{(0.7)} - \mathbf{log(0.8)} - \mathbf{log(0.2)}$$
$$= \mathbf{1.65}$$

$$\mathbf{FL(K)} = -\mathbf{0.8} * \mathbf{log(0.2)} - \mathbf{0.3} * \mathbf{log}\,\mathbf{(0.7)} - \mathbf{0.2} * \mathbf{log(0.8)} - \mathbf{0.8} * \mathbf{log(0.2)}$$
$$= \mathbf{1.184}$$

$$\mathbf{RFL(K)} = -\mathbf{log(0.2)} - \mathbf{0.3} * \mathbf{log}\,\mathbf{(0.7)} - \mathbf{0.2} * \mathbf{log(0.8)} - \mathbf{log(0.2)}$$
$$= \mathbf{1.464}$$

$$\mathbf{WH-RFL(K)} = -\mathbf{0.5} * \mathbf{log(0.2)} - \mathbf{0.3} * \mathbf{log}\,\mathbf{(0.7)} - \mathbf{0.2} * \mathbf{log(0.8)} - \mathbf{2} * \mathbf{log(0.2)}$$
$$= \mathbf{1.813}$$

Case I: Let's suppose for the first genuine transaction, its probability gets improve from 0.2 to 0.9 at next epoch i.e., (K+1) epoch, then values of loss functions become:

$$\mathbf{CEL(K+1)} = -\mathbf{log(0.9)} - \mathbf{log}\,\mathbf{(0.7)} - \mathbf{log(0.8)} - \mathbf{log(0.2)}$$
$$= \mathbf{0.997}$$

$$\mathbf{FL(K+1) = -0.1 * log(0.9) - 0.3 * log(0.7) - 0.2 * log(0.8) - 0.8 * log(0.2)}$$
$$\mathbf{= 0.63}$$

$$\mathbf{RFL(K+1) = -0.1 * log(0.9) - 0.3 * log(0.7) - 0.2 * log(0.8) - log(0.2)}$$
$$\mathbf{= 0.769}$$

$$\mathbf{WH-RFL(K+1) = -0.1 * log(0.9) - 0.3 * log(0.7) - 0.2 * log(0.8) - 2 * log(0.2)}$$
$$\mathbf{= 1.468}$$

Case II: Let's suppose the probability of the fraud transaction gets improve from 0.2 to 0.9 at (k+1) epoch, then Loss values become:

$$\mathbf{CEL(K+1) = -log(0.2) - log(0.7) - log(0.8) - log(0.9)}$$
$$\mathbf{= 0.997}$$

$$\mathbf{FL(K+1) = -0.8 * log(0.2) - 0.3 * log(0.7) - 0.2 * log(0.8) - 0.1 * log(0.9)}$$
$$\mathbf{= 0.63}$$

$$\mathbf{RFL(K+1) = -log(0.2) - 0.3 * log(0.7) - 0.2 * log(0.8) - 0.1 * log(0.9)}$$
$$\mathbf{= 0.769}$$

$$\mathbf{WH-RFL(K+1) = -0.5 * log(0.2) - 0.3 * log(0.7) - 0.2 * log(0.8) - 0.1 * log(0.9)}$$
$$\mathbf{= 0.42}$$

The values of the RFL, CEL, and FL loss functions are the same for both cases I & II as they treat both genuine and fraud transaction equally irrespective of its class. However, the fraud transaction gets more attention than the genuine transaction in WH-RFL as there was large decrease in the loss when fraud transaction was correctly classified in case II as compared to the genuine transaction in case I, since different

weight values were applied to hard genuine and hard fraud transaction.

The FL and CEL loss functions can easily overcome the class imbalance problem if they use the class weights as used in W-FL, α-FL (FL balanced by α) and W-CEL. The same has been explained as per the following calculations.

Here, $\boldsymbol{\gamma = 1}$ for W-FL and α -FL (α = 0.75 for fraud transaction and (1-α) = 0.25 for genuine transactions). The values of various loss functions at k epoch during training of the model are:

$$\mathbf{W-CEL(K)} = -\frac{\mathbf{4}}{\mathbf{3}} * \mathbf{log(0.2)} - \frac{\mathbf{4}}{\mathbf{3}} * \mathbf{log\,(0.7)} - \frac{\mathbf{4}}{\mathbf{3}} * \mathbf{log(0.8)} - \frac{\mathbf{4}}{\mathbf{1}} * \mathbf{log(0.2)}$$

$$= \mathbf{4.064}$$

$$\mathbf{W-FL(K)} = -\frac{\mathbf{4}}{\mathbf{3}} * \mathbf{0.8} * \mathbf{log(0.2)} - \frac{\mathbf{4}}{\mathbf{3}} * \mathbf{0.3} * \mathbf{log\,(0.7)} - \frac{\mathbf{4}}{\mathbf{3}} * \mathbf{0.2} * \mathbf{log(0.8)} - \frac{\mathbf{4}}{\mathbf{1}} * \mathbf{0.8} * \mathbf{log(0.2)}$$

$$= \mathbf{3.07}$$

$$\boldsymbol{\alpha - FL\,(K)} = -\mathbf{0.25} * \mathbf{0.8} * \mathbf{log(0.2)} - \mathbf{0.25} * \mathbf{0.3} * \mathbf{log\,(0.7)} - \mathbf{0.25} * \mathbf{0.2} * \mathbf{log(0.8)} - \mathbf{0.75} * \mathbf{0.8} * \mathbf{log(0.2)}$$

$$= \mathbf{0.576}$$

Case I: Let's suppose the probability of the first genuine transaction gets improve from 0.2 to 0.9 at next (k+1) epoch, then Loss values become:

$$\mathbf{W-CEL(K+1)} = -\frac{\mathbf{4}}{\mathbf{3}} * \mathbf{log(0.9)} - \frac{\mathbf{4}}{\mathbf{3}} * \mathbf{log\,(0.7)} - \frac{\mathbf{4}}{\mathbf{3}} * \mathbf{log(0.8)} - \frac{\mathbf{4}}{\mathbf{1}} * \mathbf{log(0.2)}$$

$$= \mathbf{3.193}$$

$$\mathbf{W-FL(K+1)} = -\frac{\mathbf{4}}{\mathbf{3}} * \mathbf{0.1} * \mathbf{log(0.9)} - \frac{\mathbf{4}}{\mathbf{3}} * \mathbf{0.3} * \mathbf{log\,(0.7)} - \frac{\mathbf{4}}{\mathbf{3}} * \mathbf{0.2} * \mathbf{log(0.8)} - \frac{\mathbf{4}}{\mathbf{1}} * \mathbf{0.8} * \mathbf{log(0.2)}$$

$$= \mathbf{2.331}$$

$$\alpha - \mathbf{FL}\,(\mathbf{K}+\mathbf{1}) = -\mathbf{0.25} * \mathbf{0.1} * \log(\mathbf{0.9}) - \mathbf{0.25} * \mathbf{0.3} * \log(\mathbf{0.7}) - \mathbf{0.25} * \mathbf{0.2} * \log(\mathbf{0.8}) - \mathbf{0.75} * \mathbf{0.8} * \log(\mathbf{0.2})$$

$$= \mathbf{0.437}$$

Case II: Let's suppose the probability of the fraud transaction gets improve from 0.2 to 0.9 at next (k+1) epoch, then Loss values become:

$$\mathbf{W} - \mathbf{CEL}(\mathbf{K}+\mathbf{1}) = -\frac{\mathbf{4}}{\mathbf{3}} * \log(\mathbf{0.2}) - \frac{\mathbf{4}}{\mathbf{3}} * \log(\mathbf{0.7}) - \frac{\mathbf{4}}{\mathbf{3}} * \log(\mathbf{0.8}) - \frac{\mathbf{4}}{\mathbf{1}} * \log(\mathbf{0.9})$$

$$= \mathbf{1.451}$$

$$\mathbf{W} - \mathbf{FL}(\mathbf{K}+\mathbf{1}) = -\frac{\mathbf{4}}{\mathbf{3}} * \mathbf{0.8} * \log(\mathbf{0.2}) - \frac{\mathbf{4}}{\mathbf{3}} * \mathbf{0.3} * \log(\mathbf{0.7}) - \frac{\mathbf{4}}{\mathbf{3}} * \mathbf{0.2} * \log(\mathbf{0.8}) - \frac{\mathbf{4}}{\mathbf{1}} * \mathbf{0.1} * \log(\mathbf{0.9})$$

$$= \mathbf{0.852}$$

$$\alpha - \mathbf{FL}\,(\mathbf{K}+\mathbf{1}) = -\mathbf{0.25} * \mathbf{0.8} * \log(\mathbf{0.2}) - \mathbf{0.25} * \mathbf{0.3} * \log(\mathbf{0.7}) - \mathbf{0.25} * \mathbf{0.2} * \log(\mathbf{0.8}) - \mathbf{0.75} * \mathbf{0.1} * \log(\mathbf{0.9})$$

$$= \mathbf{0.160}$$

Loss values are different for both cases for all three loss functions and in this way treat genuine and fraud transactions differently.

1.10.2.5 Threshold Moving

The techniques discussed such as resampling a data set and developing a customized loss function are used to address the class imbalance issue. However, changing the decision threshold is often overlooked. This technique can help minimize class imbalance. Changing the decision threshold can also help minimize the bias in favor of the minority positive class. Threshold moving used in the proposed methodology has been explained as the following.

A ROC curve is a graph that plots the true positive and false positive rates of predictions made by a model on an experiment data set. It uses a set of varying threshold values to interpret the true positive rate (TPR) and false positive rate (FPR) as shown in Figure 1.12. The false-positive and true-positive rates are plotted on the x-axis and the y-axis, respectively, are referred to as the ROC curve. The plot's diagonal line indicates no chance line, or we can say no-skill classifier. The ROC curve is a commonly used tool to analyze the trade-off between different threshold values.

The level of class imbalance in a data set is also an important factor that influences the decisions that a neural network can make. Due to the complexity of the data, the optimal decision threshold is very important when learning from imbalanced data [53].

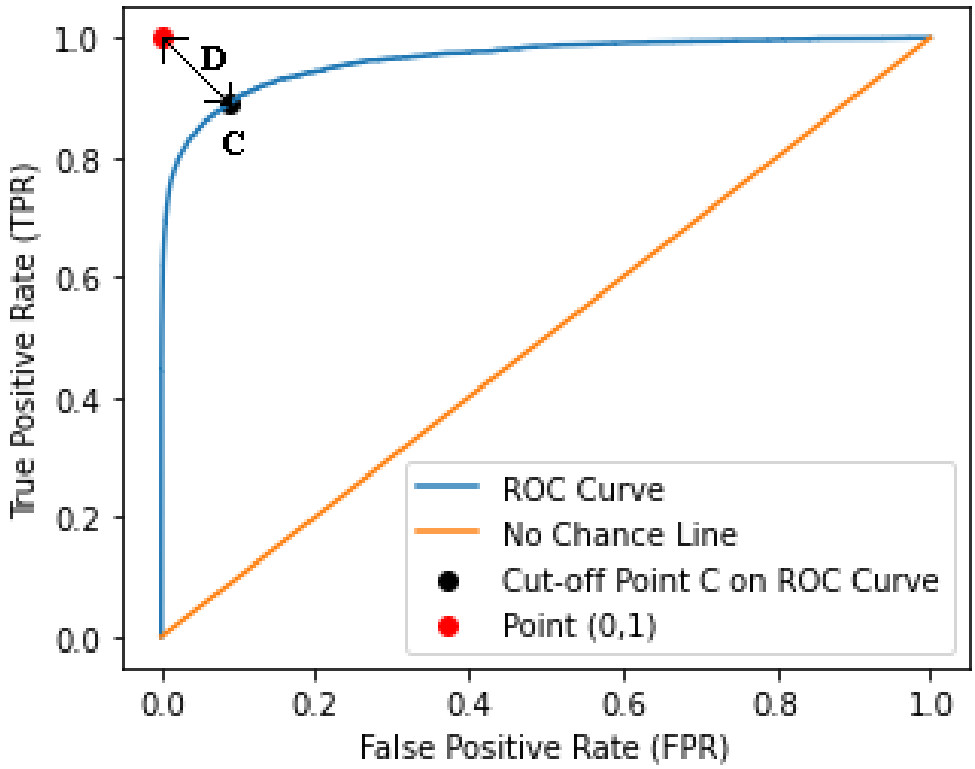

Figure 1.12: Receiver Operating Characteristic (ROC) Curve

At point (0,1), the value of FPR is zero and TPR is one and the curve closer to (0,1) point is assumed to be better. Thus, ROC curve closer to the top left-hand corner is better. The summary of the ROC curve is defined by its area under the curve which is decision threshold independent [54].

There are three main methods or criteria which can be used to select threshold from the ROC curve.

- ***Closest to (0,1) Criteria***

This is a distance calculation method [55-56] used to determine the distance between the top left corner i.e., point (0,1) to the corresponding values of false-positive and true-positive rates. Hence, distance D can be calculated as:

$$D = \sqrt{(1 - TPR)^2 + (0 - FPR)^2}$$

$$D = \sqrt{(1 - TPR)^2 + (FPR)^2}$$

$$D = \sqrt{(1 - TPR)^2 + (1 - TNR)^2} \quad (13)$$

As shown in Figure 1.12, the point 'C' on ROC curve will be selected as optimal threshold as the distance from this point to top left corner point will be minimum.

- ***Youden Index (J) Criteria***

Youden's index (J) [57] is a cut-off point whose minimum value can also be used to select the optimum cut-off point for a given ROC curve. The mathematical equation of J is:

$$J = TPR + TNR - 1 = TPR - FPR \quad (14)$$

As shown in Figure 1.13, J is the difference between the TPR and FPR. G. Kwon et al. [58] have used Youden Index (J) criteria with deep learning in their research work.

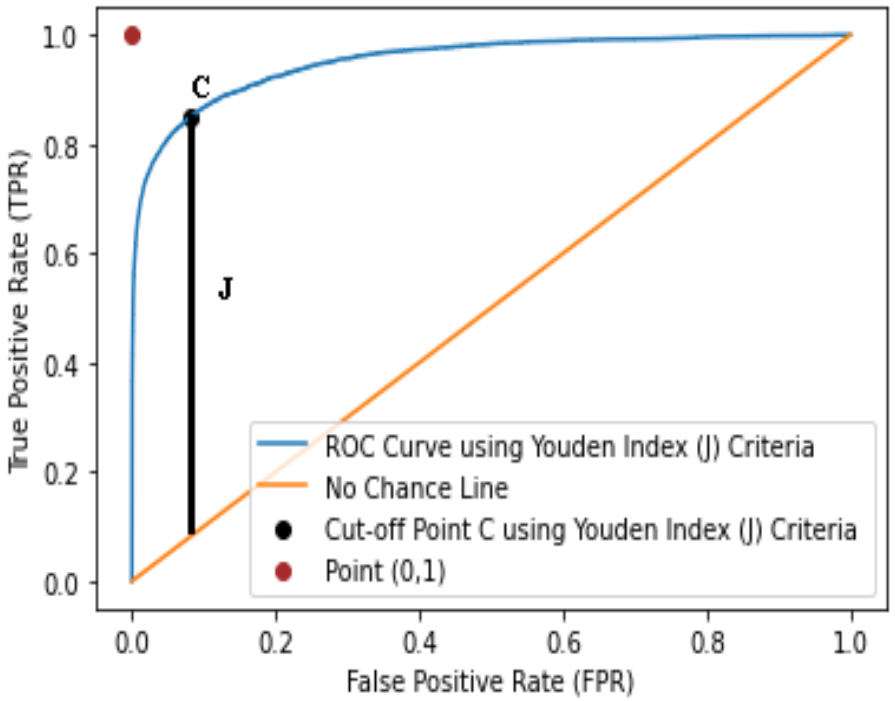

Figure 1.13: ROC Curve using Youden Index (J) Criteria

- ***Max G-Mean Criteria***

This method tries to find a cut-off point over the ROC curve where the value of G-Mean is maximum. G-Mean value can be calculated as following:

$$\mathbf{G-Mean} = \sqrt{\mathbf{TPR} * \mathbf{TNR}} \tag{15}$$

Hence, G-Mean values are calculated against the various decision thresholds to reach the C cut-off point where G-Mean value is maximum as shown in Figure 1.14.

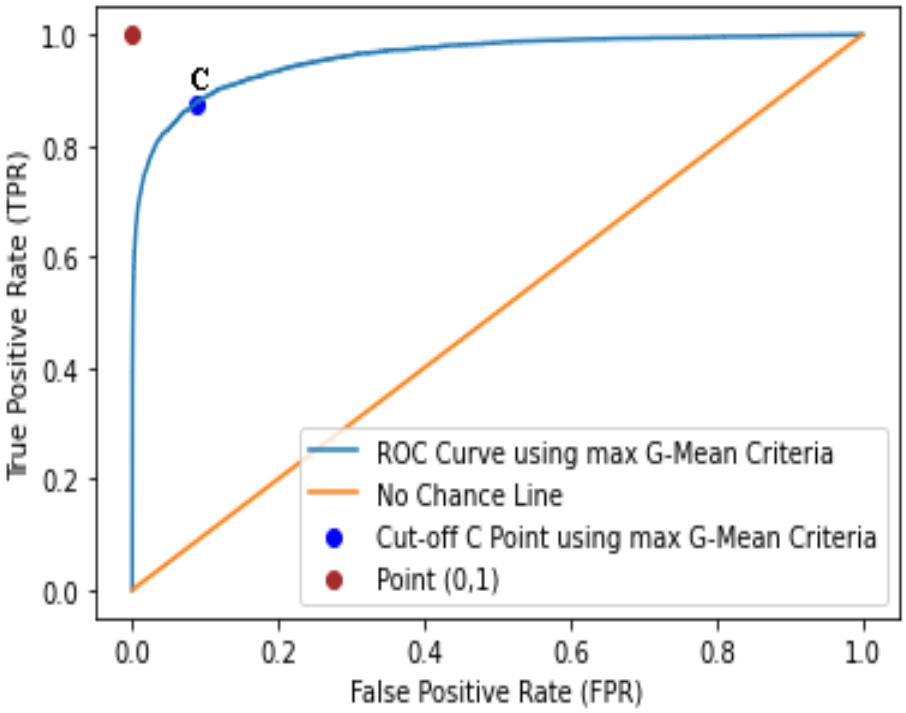

Figure 1.14: ROC Curve using max-G-Mean Criteria

1.10.3 Hybrid Methods

These methods include combination of data-level and algorithm-level methods in which they modify the dataset and adjust the learning of the model as well to handle imbalanced data.

Example: Let's suppose we have dataset containing 1000 genuine transactions and 100 fraud transactions. We perform RUS-ROS on the dataset i.e., random undersampling (RUS) on the genuine transactions and random oversampling (ROS)

on the fraud transactions. We randomly decrease the percentage of genuine transactions by 5% i.e., after random undersampling, the count of genuine transactions become 950 and we randomly increase the percentage of fraud transactions by 1% i.e., after random oversampling (ROS), the count of fraud transactions is 101. Since, the data is still imbalanced, hence threshold needs to be optimized to handle the class imbalance. Different combinations of data level methods can be used as per requirement.

1.11 Data Preprocessing Techniques

Data preprocessing is the process of preparing raw data for use with a deep learning model. It is the first and most important step in developing a deep learning model. When developing a deep learning model, data is not always come across clean and formatted before performing any operation on data, it must be cleaned and formatted. As a result, we use the data preprocessing task for this. There are various operations which are performed to preprocess the data which are as following:

1.11.1 Handling Duplicate Rows

There can be duplicate entries of the rows in the dataset. So, these need to be handled beforehand as these duplicate entries do add weightage to the class to which they belong to. In fraud detection dataset, where the dataset is already imbalanced and if there are duplicate entries for genuine transactions, it may lead to biased results. So, these duplicate rows must be identified and removed as per requirements.

1.11.2 Replacing Null Values

The important step in data preprocessing is to deal with missing values in the datasets. If our dataset contains some missing data, it may pose a significant challenge to our deep learning model. As a result, handling missing values in the dataset is required.

Methods for dealing with missing data: There are primarily two approaches to dealing with missing data. In first approach, we just delete the specific row or column which consists of null values. But this approach is not so efficient and removing data

may lead to loss of information which will not give the accurate output. Second approach is to deal with these missing or null values by imputing a particular value for these. The various methods for imputation are: i) imputation by mean value of row or column containing missing value, ii) imputing with any numerical value and adding missing value indicators in the dataset which are extra features in the dataset representing where the missing values have been imputed.

1.11.3 Categorical Data Encoding

Categorical data is divided into categories. Because machine learning or deep learning models are entirely based on mathematics and numbers, having a categorical variable in our dataset may cause problems when building the model. As a result, these categorical variables must be encoded into numbers. There are various methods to convert categorical data into numerical like one hot encoding, ordinal encoding [59] etc.

1.11.3.1 One Hot Encoding

When the features are nominal, we use this categorical data encoding technique. Nominal features do not have any order. In one hot encoding, we create a new variable for each level of a categorical feature. Each category is associated with a binary variable that contains either 0 or 1. In this case, 0 represents the absence of that category and 1 represents its presence. Dummy variables are the name given to these newly created binary features. The number of dummy variables is determined by the number of levels in the categorical variable.

1.11.3.2 Ordinal Encoding

This method considers the sequence of labels and converts them into an integer value.

1.12 Feature Selection Techniques

In machine learning, the goal of feature selection is to find out the best set of features that allows one to build learning models able to generate best results. Thus, selecting relevant features is very crucial step in building a model. Feature selection is a process that selects the most important features of a given model. This differs from feature extraction, which is a process that selects new features. Feature selection simplifies the model's development process and helps minimize storage and bandwidth consumption. There are mainly three types of feature selection techniques which are filter, wrapper, and embedded method [60].

1.12.1 Filter Method

This is a feature selection method performed by filtering the data based on its general characteristics. It is usually faster and more reliable when dealing with large datasets. Correlation, Chi-square test and information gain are most used the filter methods.

1.12.2 Wrapper Method

This is a feature selection algorithm that can be used as a wrapper around a predictive model. It provides better performance by using the same model to select features. Forward selection, backward elimination and stepwise elimination are some of the filter methods.

1.12.3 Embedded Method

This is the embedded version of the feature selection process is typically used for model building. It avoids overfitting and is less computationally expensive. LASSO, Ridge Regression, and Elastic net etc. are embedded methods.

1.13 Feature Transformation Techniques

After selecting the relevant features, various feature transformation techniques are used.

1.13.1 Feature Scaling

Feature scaling is a technique for bringing together self-variables or to normalize the feature ranges in data. Because the range of variables in the original data is very different, feature scaling is a necessary step in the stochastic gradient descent calculation [61].

1.13.2 Log Transformation

To deal with skewed data, the log-transformation is used in research works. The log transformation can reduce data variability and cause data to conform more closely to the normal distribution [62]. A log transformation analysis should be preferred over an untransformed analysis for continuous positive data measured on an interval scale. The data obtained should not require any additional justification beyond that required to support an untransformed analysis [63].

1.13.3 Standardization

The standardization technique is used to standardize values of features or attributes from different dynamic ranges into a specific range. Standardization of the all the numerical features can be performed using the z-score, min-max and decimal scaling methods [64]. The z-score standardizes the values for an attribute based on its mean and standard deviation, thus this method is useful when the actual minimum and maximum of attribute are unknown. In decimal point, the decimal point of values of an attribute is moved to standardize it and the number of decimal points moved depends on the maximum absolute value of the feature. The min-max transforms the data set between 0.0 and 1.0 by removing the minimum value from each value and dividing the result by the range of values for each individual value.

1.14 Outline

In chapter 1, the problem of fraud detection, types of online frauds, reasons for their cause and their preventive measures have been discussed. Then, parameters of fraud detection have been explained. The role of deep learning in fraud detection has also been discussed. The problem of class imbalance and methods to handle the same has been presented. In chapter 2, existing research work using deep learning for fraud detection has been reviewed and the various research gaps have been identified. In

chapter 3, research hypotheses and objectives have been explained. In chapter 4, data preprocessing and feature selection techniques have been discussed for each dataset. In chapter 5, a system proposed for online transaction fraud detection has been presented. In chapter 6, the comparison of proposed fraud detection system with existing machine learning based fraud detection systems has been done. In chapter 7, the results of proposed system along with the existing systems have been summarized. In chapter 8, the conclusion of the proposed research work has been presented and the future scope has also been discussed.

Chapter-2

Literature Review

2.1 Reviewing Existing Research

Deep learning takes the advantage of the multiple hidden layers in the data. It is usually used for learning systems that are based on different types of data and methodologies. The literature review section aims to provide a comprehensive review of the various research findings related to deep learning in fraud detection.

U. Fiore et al. [65] have proposed a framework for detecting credit card frauds from imbalanced dataset. Since the dataset used is highly imbalance in nature, oversampling the minority class transactions is one of solution but with some disadvantages also. For the study, the researchers trained a GAN to mimic fraud transactions which were later used along with the training data. They first trained a classifier on original dataset and performed hyperparameter tuning of the classifier and produced best results for the test data. They separated the fraudulent transactions from the dataset and used these fraud transactions as input training set to the GAN and performed its hyperparameter tuning. Then, they used GAN for generating synthetic fraud transactions. They combined the generated synthetic fraud transactions with the original training set. Then, they again trained the classifier on this extended dataset to get the results for test data. They have also used Synthetic Minority Over-Sampling (SMOTE) technique to generate synthetic fraud transactions and compared the results with the GAN which were not better. To validate their results, they have used publicly available Credit Card dataset which is highly imbalanced and contains very few fraudulent transactions as compared to the genuine transactions. From their experiment results, they have found that the by using synthetic fraud transactions, the sensitivity has been increased and the number of false positives has also been increased.

Z. Zhang et al. [66] have proposed fraud detection model using a Convolutional neural network (CNN) which contains a feature sequencing layer that reorganizes the features of transactions to form various convolutional patterns. The benefit of which is that various derivative features will be produced by different convolutional patterns. Apart from feature sequencing later, the network contains four convolutional layers and pooling layers and one fully connected output layer. The model proposed by them is efficient for low dimensional and non-derivative features data. For their experimentation and validation of proposed model, they then extracted five million transactions data and compared it to the data of the minority groups. They performed sampling to balance data. They have used 8 dimensional features as input for their model. Their proposed model performed better than the existing CNN model and BP neural network in terms of both Precision and Recall by improvement of 26% and 2% in their values respectively.

J. A. Gomez et al. [67] have used 2 different datasets containing fraud claims and two anonymized transactions. The researchers have implemented a cascade of two filters that consists of every single neural network being trained as a binary classifier. The researchers were able to reduce the genuine to fraud ratio to 420:1 by using two filters. They have evaluated the performance of their system by using value detection rate and true false positive rate. Value detection rate is the amount of money involved in frauds detected by the system. True false positive rate is the total number of transactions predicted as fraud to the actual fraud transactions detected by the system. They have also used area under ROC curve. After filtering data, they used multi-layer perceptron to detect whether the transaction is genuine or fraud. The experiment results have shown that the proposed system produces better results than existing techniques as validated on the real data.

J. Jurgovsky et al. [68] have proposed a fraud detection system based on a sequence learner i.e., LSTM (Long Short-Term Memory) and compared the results with the static learner i.e., RF (Random Forest). Feature extraction along with manual feature aggregation techniques have been utilized in their research work. They have also performed under sampling to identify fraudulent and real accounts. There are one

million accounts in training data. The researchers reduced samples of both classes to bring their ratio of 1:10 for fraud and genuine transactions. Area under Precision-Recall curve has been used to evaluate the performance of the proposed system. From their research work, they have found that manual aggregation improves the performance of both sequence (LSTM) and static (RF) classifier on offline and online transactions. For offline transactions, LSTM is better than RF. They have also found that the frauds detected by both LSTM and RF are different and hence, the results of both can be combined for better results.

Chouiekh and E. H. I. E. Haj [69] have used Deep Convolutional Neural Network (DCNN) for fraud detection purposes. The real mobile communication dataset has been used in their research work. The dataset was in the form of artificial images indicating user's transactional history. Any deviation in history indicates the suspicious activity from the user's side. 18000 images for 300 users were being used for 60 days' period of user's activity. Random Search Procedure has been utilized for tuning hyperparameters of the DCNN model. They have compared the results with state of art methods i.e., Gradient Boosting Classifier (GBC), Support Vector Machine (SVM), and Random Forest (RF). They have used k-fold cross validation technique to avoid overfitting of the models. The accuracy of DCNN models was observed for different number of epochs and the training time of all models was also observed for different sizes of training data. DCNN model performed better than all other models in both computation time and accuracy.

Pumsirirat and L. Yan [70] proposed an Autoencoder (AE) and restricted Boltzmann machine (RBM) based unsupervised fraud detection system which is capable to detect the frauds from the patterns of normal transactions. As the fraudster keep changing the patterns with time, it has become more challenging to detect frauds due to these unstable patterns. Hence, they have focused their research work on unsupervised learning as compared to supervised learning as in unsupervised learning, suspicious patterns are identified from the normal transactions and then in real time, these patterns are used to detect the fraudulent transaction. The various performance metrics used are area under Receiver Operating Characteristics (ROC) curve, root

mean squared error and mean squared error. Total three datasets have been used. First one is German dataset having 1000 transactions, second one is Australian dataset having 690 transactions and the third one is European dataset having 284,807 transactions. They have achieved maximum AUC score for larger dataset i.e., European for both AE and RBM i.e., 0.9603 and 0.9505 respectively as compared to other two datasets.

N. Abroyan [71] proposed CNN and LSTM-RNN based financial market risk classification model. The author has used a real-world dataset of risky transactions to train his Deep Learning system. The training set had over 10,000 samples and 250 features. The dataset is balanced as it contains equal number of fraud and genuine transactions. The one-dimensional convolutional neural network (CNN) contains one convolution-pooling layer and one hidden layer from full connected layer. LSTM contains two hidden layers. K-fold cross validation has been used for hyperparameter tuning of the models. The loss function used is binary cross entropy (BCE). In hidden layers, Rectified Linear Unit has been used and the sigmoid function has been used in the output layer as fraud detection is a binary classification problem. Dropout has been used for regularization. The performance metric used is F1 score which acts as the tradeoff between the precision and recall. CNN achieved F1 score having value of 0.8 and LSTM achieved 0.91 F1 score better than the results produced by state of arts methods. Hence, LSTM has achieved the best results among all other models.

Wiese and C. Omlin [72] have proposed a fraud detection system which is able to learn sequences from the transactions. They have used Long Short-Term Memory-Recurrent Neural Network (LSTM-RNN) for this purpose as they believe that finding the sequences from the transactions is more useful than the individual transactions. Support Vector Machine (SVM) has also been implemented for comparison basis. They have used real dataset of more than 30,000 transactions performed in a year. Preprocessing techniques and feature selection has been performed on the dataset before feeding to the classification models. They performed the under sampling on the dataset to reduce its size for their experiments and to bring

the ratio of genuine to fraud transactions as 99:1 i.e.., for 99 genuine transactions, there is one fraud transaction. The classification rate achieved by LSTM is 2690 transactions per second and Max AUC and Average AUC scores achieved are 0.99216 and 0.98221 respectively. On the other hand, SVM achieved classification rate of 213 transactions per second, Max AUC score of 0.89322.

K. Fu et al. [73] have designed a fraud detection system for finding hidden patterns from the data by utilizing CNN. The system includes feature extraction, data sampling and feature transformation modules. Training is performed using historical transactions and the test data is an online transaction. Hence, training part is offline whereas test part is online. A novel feature named trading entropy has also been proposed by them which helps in detecting more complex features. A real dataset provided by a commercial has been used. Cost-based sampling improved the system's performance by generating synthetic fraud samples and on the other hand, under-sampling was performed on genuine transactions. The results have been compared with various state of art methods which are Random Forest (RF), Neural Network (NN), and Support Vector Machine (SVM). Maximum F1 score achieved is achieved by CNN i.e., approximately 3.4.

S. Wang et al. [74] proposed a deep learning (DL) based fraud detection system named CLUE. The proposed system can capture detailed information regarding the user's actions and the sequences of the actions performed is also modelled using Recurrent Neural Network (RNN). They have also applied data and algorithm-level approaches to address imbalanced data. For their study, under sampling was performed to get a dataset of more than one million sessions. They then applied cost-sensitive learning to their model to minimize the misclassification cost. The proposed system's performance was compared to the existing machine learning and Fully Connected- Neural Network (FC-NN). FC-NN performs better than the traditional machine learning algorithms and RNN performs better than FC-NN as it can capture sequence information about user actions and enable to use embeddings in the model we well. Also, deep learning is well suited for large amount of data as compared to the state of art machine learning methods. Hence, the

proposed system is much efficient and require no additional feature engineering to perform with RNN.

A. Shen et al. [75] have proposed a framework to choose suitable model for credit card fraud detection by comparing transactions with the historical data. They have implemented three learning models i.e., Neural Network (NN), Decision Tree (DT) and Logistic Regression (LR) based classification models for fraud detection. The real transactional dataset was used containing more than forty features. The data was labelled as fraud or non-fraud. The fraud transactions are very less in number in the dataset as they are only 0.07% of whole data. Hence, they performed sampling to reduce the frequency of genuine transactions in the training dataset to balance it to some level. They performed preprocessing the dataset. Missing values were excluded from the dataset. Log transformation and standardization were performed to derive features. Then, feature extraction and feature selection methods have been performed on the derive features. From the experiments, it has been found that LR and NN's performance is better than DT in detecting credit card frauds. NN performs better than the LR as well.

E. L. Paula at al. [76] proposed an unsupervised deep learning autoencoder to detect fraud for exports and money laundering. The dataset used is related to exports of products occurred in Brazil. They have performed data preprocessing techniques and feature selection. Mean Squared Error (MSE) has been used a measure to calculate the deviation of predictions from the real input data. The results have been compared to Principal Component Analysis (PCA) and it shows that the performance achieved using Autoencoder is 20 times better than the PCA as Autoencoder also detects even indistinct frauds which PCA is not able to do so and needs much computational power. PCA requires all training data to reduce the dimensionality which makes it more computationally expensive than the Autoencoder. Autoencoder can detect from frauds from the high dimensional features data as well which is linear PCA is unable to do so. Hence, Autoencoder is much accurate in detecting frauds in exports and has a better nonlinear generalization.

Z. Kazemi and H. Zarrabi [77] proposed a Autoencoder based credit card fraud detection system. The dataset used is German Credit Card dataset in which 900 samples were utilized as training and 100 as test data. Number of features used were 20. At very first stage of the proposed system, input features were passed to initial layer of autoencoder. Autoencoder was trained having 3 layers to extract the important features from the input data. After training, a softmax classifier is appended to autoencoder to obtain the class labels of the data. The results have been compared with various state of art methods which are as K-Nearest Neighbour (KNN), Genetic Programming (GP), Self-Organization Map (SOM). The overcomplete Autoencoder based system achieved maximum fraud detection rate of 84.1 with very low variance as compared to other systems.

N. Abroyan [78] proposed a deep learning-based framework for classification real time nonstationary transactional fraud data. The author implemented one dimensional Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM)- Recurrent Neural Network (RNN). The dataset used German Credit Dataset. K-fold cross validation has been used as the dataset is small and K=10 has been used. Loss function used is Binary Cross Entropy (BCE) and in hidden layers, Rectified Linear Unit function has been used. Sigmoid activation function has been used in the output layer as fraud detection is a binary classification problem. Dropout rate of 0.5 has been used for regularization and F1 score has been used for measuring the performance of the models as it acts as balancing factor between precision and recall score. CNN achieved F1 score of 0.8 and LSTM-RNN achieved 0.92 F1 score. Thus, LSTM is better than CNN as it considers the previous transactions for better prediction, and it performs better than various other state of art methods. LSTM having same architecture was trained on other credit card fraud detection dataset for which it achieved F1 score of 0.91. Thus, LSTM proved to be an efficient method for classification of non-stational fraud detection data.

Y. Heryadi and H. L. H. S. Warnars [79] have used Convolutional Neural Network (CNN), Stacked Long Short-Term Memory (LSTM), and combined CNN-LSTM models for learning temporal representation of transaction features.

Indonesia bank dataset has been used. Under sampling method was utilized to balance the majority class with respect to minority class. The researchers then generated four datasets that have different ratios of non-fraudulent and fraudulent transactions. AUROC score has been used to evaluate the performance of the learning models. The experimental results have shown that higher the non-fraud to fraud transactions ratio, higher is the accuracy. The accuracy performance metric is not suitable for imbalanced dataset. Principal Component Analysis (PCA) has been used to select the principal components from the dataset. Hence, the models have been trained on 20, 30, and 40 derived principal components. CNN performs better than CNN-LSTM and Stacked LSTM. CNN-LSTM performs better than Stacked LSTM. Hence, CNN has proved to be efficient and robust among all models in research work.

X. Lp et al. [80] have proposed a Recurrent Neural Network (RNN) based fraud detection model. They have combined ensemble and Deep Learning models to propose a sandwich shaped architecture. The data used is UnionPay transactional dataset of three months period. They also performed sampling on the dataset to balance the ration of fraud to non-fraud transactions. In very first stage, the authors performed feature engineering and then, Gradient Boost Decision Tree (GBDT) has been used for optimizing the features within a single transaction. Gated Recurrent Unit (GRU) was applied to the transformed features to learn the relationships between the transactions. In last, Random Forest classifier (RF) was trained using the optimized transaction eigenvectors. From the experimental results, they have shown that their proposed model architecture within-between-within (WBW) is better than within-within-between (WWB) and between-within-within (BWW) architecture. Hence, the proposed architecture can be implemented in the production very efficiently.

C. Wang et al. [81] have proposed Back Propagation Neural Network (BP-NN) optimized with whale algorithm used for detection of credit card fraud detection. The whale algorithm is used for optimizing the performer of BP-NN in terms of slow convergence rate and to get trap into local minima. The dataset used is

very imbalanced as it contains a very few numbers of fraud transactions. The researchers then randomly selected the samples for training and test data. For both training and the test sets, the data samples were randomly selected. The proposed system's comparison has been done with the remaining models i.e., GA-BP and PSO-BP. Whale optimization algorithm (WOA) improves the convergence speed of neural network by making it faster and helps it to achieve the global minimum by optimizing its weights. Maximum Fraud Detection rate achieved by proposed system on a test set is 98.04%.

P. Zheng et al. [82] have trained adversarial networks to detect fraud using only real users. The frauds are detected using the discriminator of complementary GAN by training it. Random under sampling was performed by them on two datasets. The researchers then split the test data into two batches. The first batch had same ratio of genuine and fraud transactions and other one having imbalanced ratio of genuine transactions and fraud transactions having 10:1 ratio. Results of proposed model have been compared with existing models and have been checked using raw features and represented features as well. Accuracy achieved by OCAN is maximum than other state pf art methods on both types of training data. Thus, it detects frauds early as it takes the sequences of transactions in account as well. OCAN only uses genuine users' data during the training part, on the other hand M-LSTM requires both the genuine and fraud users' data while training.

Y. Ando et al. [83] have proposed a Recurrent Neural Network (RNN) based fraud detection system with and without LSTM (Long Short-Term Memory). Their proposed system detects frauds by learning the behavioral patterns from the web access logs. The researchers then randomly selected the web access logs of the normal users as compared to the malicious users. Input layer units in learning model i.e., Recurrent Neural Network (RNN) were determined as per the hostnames present in positive class data and a threshold value of 0.5 was set for error loss to stop further training. The results have been compared with Support Vector Machine (SVM) using linear kernel and RBF kernel. F-Measure score achieved by RNN with and without LSTM is maximum i.e., 1 as compared to SVM with linear kernel and RBF kernel

which achieved F-Measure score of 0.5967 and 0.5844 respectively. Also, it has been noted that LSTM-RNN's convergence rate is faster than RNN without LSTM. Thus, RNN is better than state of art machine learning algorithm i.e., SVM.

Y. Lu [84] has proposed a fraud detection system based on deep learning. European dataset was used by the author. Re-sampling has been utilized on an imbalanced training data. Oversampling was then applied to the training set to make sure that the minority class ratio is same as that of majority class. Synthetic Minority Over-sampling Technique (SMOTE) was used to create synthetic examples of minority class which ultimately improved the model's performance on the test data. Logistic Regression (LR) has been used as benchmark and it has achieved 96.04 % recall. With oversampling, accuracy achieved by DNN is 96.34 % and Recall is 94.06%. Thus, the performance of neural got network increased after resampling the unbalanced dataset to make it balance rather than increasing hidden layers.

M. Renström and T. Holmsten [85] proposed an Autoencoder based fraud detection model to detect frauds from unlabelled data. They have implemented three types of Autoencoders, simple single autoencoder with three layers as baseline, Stacked Autoencoder and variational autoencoder. In stacked autoencoder, multiple autoencoders were individually trained and then stacked together. The researchers then used the two datasets to evaluate their findings. The first one was the credit card database, and the second one was the interaction with ads. Various performance metrics have been used to evaluate the performance of the models like precision, recall, accuracy, and non-predicted value (NPV). From experimental results, it has been found that Stacked autoencoder outperformed the single and variational autoencoder, hence able to detect frauds efficiently.

M. Schreyer et al. [86] have proposed a first autoencoder based system for detecting anomalies in large accounting data of journals. In their research work, they have demonstrated that the reconstruction error of the trained deep autoencoder networks can be efficiently used as an assessment tool for detecting anomalies of journals from the accounting data. They have used two ERP datasets in which dataset

A contains 307457 journal entries and dataset B contains 172990 journal entries. The proposed model has been compared to various state of art models which are Principal Component Analysis (PCA), one class Support Vector Machine (SVM), Local Outlier Factor (LOF) and Density Based Spatial Clustering of Applications with Noise (DBSCAN). Grid search method has been utilized to optimize model's hyperparameters. Their proposed model achieved superior performance than other models. It has achieved F1-score of 32.93 for dataset A and 16.95 for dataset B. Also, false positives reported were also very less than other baseline models used in the research work.

H. Choi et al. [87] have proposed a deep learning-based fraud detection model. They have also proposed generative ensembles which are independent of models for out of distribution anomaly detection using density-based method and uncertainty estimation. The dataset used for fraud detection contains fraudulent and genuine transactions. The models used for comparison were Watanabe Akaike Information Criterion (WAIC), Single Importance Weighted Autoencoder (IWAE). Deep neural network with two hidden layers has been implemented which was trained on both normal and fraud transactions whereas other two models were used only normal transactions. False Positive Rate (FPR) with fixed True Positive Rate (TPR=95%), Area under Receiver Operating Characteristics (AUROC) curve and average Precision have been used for performance analysis of system. AUROC score achieved is 99.1 and average precision score achieve is 99.3. Also, very less false positives reported i.e., FPR achieved is 4%.

J. Johnson and T. Khoshgoftaar [53] have proposed a fraud detection system by utilizing various data-level and algorithm-level techniques for handling class imbalance in Medicare fraud data. They have utilized deep neural network (DNN) in their proposed system. Threshold of DNN was optimized by them using Max G-Mean criteria and various loss functions were used with deep learning model. They have applied algorithm level methods i.e., loss functions like focal loss and mean false error along with data-level methods which are random under sampling (RUS), random oversampling (ROS), and hybrid of both previous mentioned samplings i.e., ROS-

RUS. They have used thresholding to generate decision threshold from validation data. The ROS and ROS-RUS have achieved superior performance among data level as well as algorithm-level methods and got AUC score of 0.8505 and 0.8509 respectively. ROS-RUS has also got very less training time and thus making the model four times more efficient in terms of execution speed. Algorithm level methods are more efficient in achieving stable decision boundaries among classes as compared to data level methods. In their experiments, they have observed a strong linear relationship between the optimized decision threshold and minority class size. They have also found that the minority class size and decision threshold optimized using validation data are related to each other.

F. Ghobadi and M. Rohani [88] proposed a cost-sensitive neural network (CSNN) that can be used for fraud detection by altering its backpropagation learning. A meta cost of classification has been presented by them which is the sum of the learning costs that are computed by various models trained on replicating datasets. They have used a real dataset from Brazilian credit card issuer which contains fraud and genuine transactions in an imbalance manner. The results have been compared to baseline methods which are as neural network (NN) and artificial immune system-based fraud detection model (AFDM). The proposed CSNN based fraud detection model achieved better results in terms of fraud detection rate and decreased the loss occurred due to frauds.

T. T. Nguyen et al. [89] have utilized CNN and LSTM for their work. Three datasets were used by them, and all were imbalanced. Hence, they use data -level methods like undersampling and SMOTE for making datasets balance. The results of the experiment revealed that the proposed solutions performed better than the traditional models. One of them was the LSTM with 50 blocks, which had an F1-Score of 84.85%. The sampling methods used in the experiment enhanced the performance on the existing data. However, the results of the unseen data were affected badly.

R. Y. Gupta et al. [90] have used three data imbalance techniques and six

neural network models. They were able to achieve their goals through the use of data collected from India's massive universal health coverage program known as PM-JAY. Three sampling techniques used by them were SMOTE, adaptive synthetic sampling and tabular Generative Adversarial Networks. They performed oversampling by generating duplicate transactions for the minority class i.e., fraud claims. For performance analysis, they have used F1-Score, Recall, Precision, Accuracy, and AUC-ROC scores. Neural Network trained on under sampled data performed superior to the rest of the models.

2.2 Literature Review Summary

Most of the research works discussed so far in this literature review have tried to balance their data by using data-level methods. A very few research works have handled the class imbalance problem by using algorithm-level methods or hybrid methods. Some have not revealed the methods used to handle class imbalance. The Figure 2.1 shows the various class imbalance methods that are commonly used in the field of fraud detection. Most of the studies in this area mainly utilized data-level techniques. There have been a few studies on the use of algorithm-level class imbalance methods.

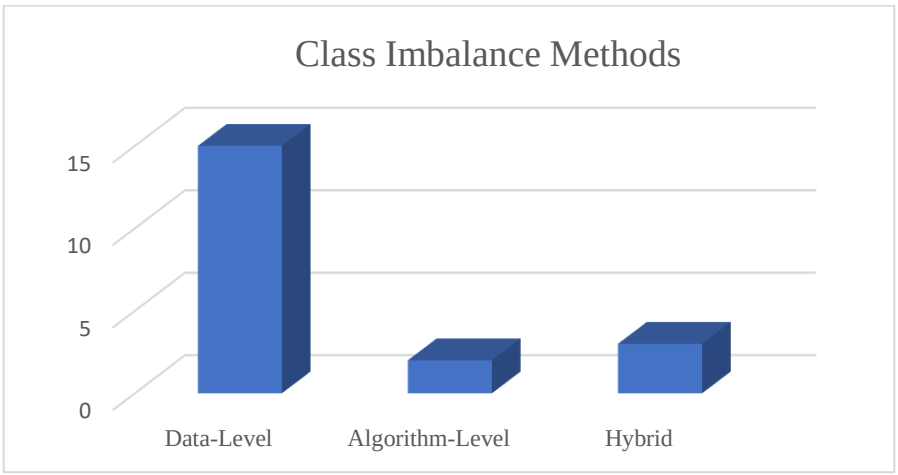

Figure 2.1. Class imbalance methods used in DL based fraud detection systems

Table 2.1

Overview of Literature Review

Type of Fraud Detection System Title of Research Work	**Year of Publication**	**Indexing of journal**	**Method/Dataset/Conclusion**	**Metric used/ Remarks**
Credit Card Fraud Detection System [65]	2017	SCI	• GAN based credit card fraud detection model • European dataset • Data-level methods were used to handle class imbalance	TPR improved at small decrease in TNR
Online Transaction Fraud Detection System [66]	2018	SCI	• CNN based online transaction fraud model • This study was carried out using a commercial bank's database, which contained over 5 million business to customers transactions.	Precision: 91% Recall: 94%
System for fraud scoring in card payments [67]	2018	SCI	• Deep Neural Network Based fraud scoring model in card payments • The data collected for this study came from Jan 2014 to June 2015, and it included 900 million plus transactions.	ROC results
Credit-card fraud detection System [68]	2018	SCI	• LSTM-RNN and Random Forest based model credit card fraud detection model • The data was collected from March to May 2015. It included real credit card transactions.	Manual feature aggregation improved the performance of both LSTM and RF.

Fraud Detection System [69]	2018	Scopus	• CNN based fraud detection model • The model was developed based on the customer details record dataset of a mobile communication carrier.	Accuracy (LSTM) is highest i.e., 82% and training time is also less as compared to SVM, RF, GBC.
Credit Card Fraud Detection System [70]	2018	ESCI	• Autoencoder and RBM based system • The model was developed using three datasets: a German database, an Australian database, and a European one.	Area Under Curve
System for Financial Market Risk Classification [71]	2017	CNKI Scholar	• CNN and LSTM-RNN based financial market risk classification model • The model was developed based on a real balanced dataset.	F1 Score
Credit Card Fraud Detection System [72]	2009	Part of the Studies in Computational Intelligence (SCI) book series, Springer	• LSTM-RNN and SVM based credit card fraud detection model • The real-world transaction data collected in a year showed that there were over 30,000 transactions.	Area Under Curve (AUC) Score
Credit Card Fraud Detection System [73]	2016	Part of Lecture Notes in Computer Science LNCS, Springer	• CNN based system • The real transaction data that was used for the study came from a commercial bank, and it contained over 260 million transactions.	F1 Score
Online E-Commerce Transactions Fraud Detection System [74]	2017	Part of Lecture Notes in Computer	• LSTM-RNN based model for online transaction fraud detection	RNN performed better than FC-NN

		Science LNCS, Springer	• The production data that were used for the study came from web server logs.	
Credit Card Fraud Detection System [75]	2007	IEEE Conference proceedings	• Neural Network (NN), Decision Tree (DT) and Logistic Regression (LR) based fraud detection system • The database that contained the transaction history included over 40 fields.	Neural network outperforms as compared to machine learning models
Fraud Investigation System in Exports and Anti-Money Laundering [76]	2016	IEEE Conference proceedings	• Autoencoder based fraud detection model for exports and anti-money laundering • Exports data of goods and services in Brazil occurred in 2014.	Good Performance achieved
Credit Card Fraud Detection System [77]	2017	IEEE Conference proceedings	• Autoencoder based credit card fraud detection model • German Credit Dataset.	Fraud Detection rate: 84.1
System for real-time data classification for credit transactions [78]	2017	IEEE Conference proceedings	• CNN and LSTM-RNN based model for credit data classification • German Credit Dataset	F1 Score
Fraudulent transaction Detection System [79]	2017	IEEE Conference proceedings	• Stacked LSTM, CNN and CNN-LSTM based models for learning temporal representation of transaction features • The dataset included 50 features from an Indonesia bank in 2016-2017.	AUC Score

Transaction Fraud Detection System [80]	2018	IEEE Conference proceedings	• RNN based fraud detection model • Three months period transactional data was used.	Good performance with imbalanced data.
Credit Card Fraud Detection System [81]	2018	IEEE Conference proceedings	• Back Propagation Neural Network optimized with whale algorithm used for detection of credit card fraud detection. • Credit Card Transactions Data	Fraud Detection Rate: 98.04%
Fraud Detection System [82]	2018	AAAI Conference Proceedings	• Autoencoder and complementary GAN based transactional fraud detection model. • European dataset	Maximum Fraud Detection accuracy
System for Detecting Fraudulent Behavior on web [83]	2016	Computer Security Symposium Proceedings	• RNN based model for detecting fraudulent behavior with and without LSTM • The dataset included web access logs from Yahoo.	F-Measure (with and without LSTM): 1
Fraud Detection System [84]	2017	Independent Thesis Advanced Level published in DiVA	• Deep Neural Network Based Model for fraud detection • European cardholders dataset was used.	Accuracy: 96.34% Recall: 94.06%
Unsupervised Fraud Detection System [85]	2018	Independent Thesis Published in DiVA	• Autoencoder based fraud detection model using unlabeled data • Two different datasets were used.	Stacked autoencoder outperforms the single and variational autoencoder
System for Anomalies Detection in Large Scale Accounting Data [86]	2018	Paper published in arXiv.org	• Autoencoder based model for detecting anomalies in accounting data • The SAP ERP	F1 Score

			datasets included journal entries.	
Anomaly Detection System [87]	2019	Paper published in arXiv.org	• DNN based fraud detection model • Fraud Transactions Dataset	Area Under ROC curve (AUROC)
Medicare Fraud Detection system [53]	2019	SCI	• DNN based fraud detection system • Algorithm-level and data-level methods have been utilized for handling imbalance data • Medicare datasets	ROC AUC Scores: 0.8505 and 0.8509 for ROS and ROS-RUS respectively.
Credit Card Fraud Detection [88]	2016	Conference Proceedings	• Neural Network based fraud detection system using Cost Sensitive method • Brazilian Credit Card Dataset containing genuine and fraud transactions and imbalance in nature	Better fraud detection rate and decreased loss due to frauds.
Credit Card Fraud Detection [89]	2020	Paper published in arXiv.org	• CNN and LSTM based fraud detection system • Data-level methods applied on three datasets to balance them	F1 score
Fraud Detection System for health coverage schemes [90]	2021	Scopus	• DNN based fraud detection system • Data-level methods applied on a healthcare data containing more than 3 lakh claim records	F1 score and various other metrics were used.

2.2.1 Research Gaps

Due to the complexity of the work involved in developing fraud detection systems, many researchers have been unable to provide detailed information about their projects. Fraud detection is a flourishing field of research that focuses on studying new techniques to prevent frauds. From the literature review discussed, it is evident that there are many research gaps or challenges in detecting online transactions fraud, some of which are as the following:

- One of the main challenges in detecting online fraud is the lack of a good dataset. Many financial institutions don't allow the use of their data due to privacy issues.

- Due to lack of dataset availability, researchers have used synthetically generated datasets which affects the system performance. Also, the features selected, and results achieved by them are not completely revealed which leads to difficulty in future research.

- Fraud patterns are not static as they keep on changing over time. Rule-based systems are incapable to deal with the changing fraud patterns.

- Pattern matching is also a major challenge faced by the fraud detection system because some fraud patterns look similar to normal patterns. The research work will try to detect these fraudulent patterns efficiently using deep learning.

- The transactional datasets are huge in size as they may contain millions of transactions which makes the fraud detection process more complex for the existing techniques applied so far. Deep learning techniques are capable to deal with huge datasets.

- Class imbalance is inherent transactional data. To handle such imbalanced data, deep learning has been considered as the most effective method for

detecting fraud. Also, many researchers have used data level techniques that alter the dataset to handle the class imbalance method. Changing the data set's composition can prevent the important information from getting lost. This research work aims to propose a fraud detection system by utilizing study the algorithm-level techniques that can handle the class imbalance problem without modifying the data.

Chapter-3

Research Hypothesis and Objectives

The goal of this research work is to develop an efficient and intelligent system that can detect fraudulent transactions in real-time. This system has utilized deep learning techniques to detect the most number of fraudulent transactions. The classification of fraudulent and genuine transactions is typically a binary task. However, there can be more than two types of transactions. Thus, to classify an incoming online transaction whether it is fraud or not, various steps are performed in a fraud detection system as shown in Figure 3.1.

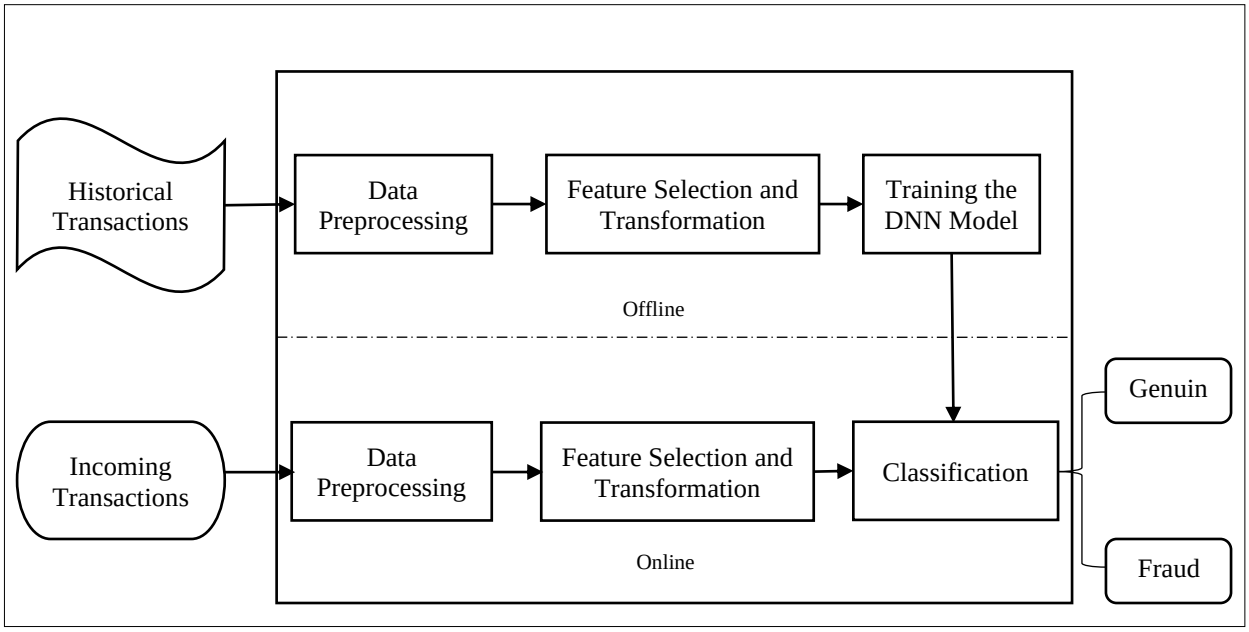

Figure 3.1: Online Transaction Fraud Detection System

The fraud detection system consists of two components: the offline training component, which is used for learning the DNN model, and the online component, which is used for detecting fraudulent transactions.

Performance metrics are used to measure the performance of the system. Selection of a suitable performance metric for measuring a system's performance is very important. Confusion Matrix [91] is a 2x2 table (Table 3.1) which contains count values of false negatives (FN), false positive (FP), true positives (TP), and true negatives (TN). The positive class in our case includes fraud transactions and negative class includes genuine transactions. Hence, various metrics can be computed from these counts.

Table 3.1

Confusion Matrix for our fraud detection system

		Predicted Value	
		Positive (Fraud)	**Negative (Genuine)**
True Value	**Positive (Fraud)**	TP	FN
	Negative (Genuine)	FP	TN

True Positive Rate (TPR) or Recall

The TPR of the system is the percentage of frauds transactions that the system correctly predicts.

$$\mathbf{TPR} = \frac{\mathbf{TP}}{\mathbf{TP} + \mathbf{FN}} \tag{16}$$

True Negative Rate (TNR)

The TNR of the system is the percentage of genuine transactions that the system correctly predicts.

$$\mathbf{TNR} = \frac{\mathbf{TN}}{\mathbf{TN} + \mathbf{FP}} \tag{17}$$

False Positive Rate (FPR)

FPR is the percentage of genuine transactions that the system wrongly identifies as fraud transactions.

$$\mathbf{FPR} = \frac{\mathbf{FP}}{\mathbf{FP} + \mathbf{TN}} = 1 - \mathbf{TNR} \tag{18}$$

Accuracy

It measures the degree of closeness of predicted value of a transaction to the actual value of transaction. In class imbalanced scenario, accuracy is not a good performance metric as it can be biased to the majority class.

$$\mathbf{Accuracy} = \frac{\mathbf{TP} + \mathbf{TN}}{\mathbf{TP} + \mathbf{FN} + \mathbf{TN} + \mathbf{FP}} \tag{19}$$

Geometric Mean (G-Mean)

As per the name suggest, it is the geometric mean of TPR and TNR. The equation (15) of G-Mean is already discussed.

All above discussed performance metrics are calculated using a threshold value of the learning model. The ROC curve has been utilized for selecting optimal threshold for these performance metrics.

Receiver Operating Characteristic (ROC) Curve

The curve plotted between the True Positive Rate (TPR) and the False Positive Rate (FPR) is known as the ROC curve [92]. Area under the Receiver Operating Characteristic Curve (AUROC) is used to evaluate the performance of the models and it is decision threshold independent. The decision threshold for this study was optimized using the Closest to (0,1) method. This method gives equal importance to the TPR and TNR than other thresholding criteria [93].

Thus, in this research study all the above-mentioned metrics will be used to measure performance of the fraud detection system.

Our Proposed framework for fraud detection model is also applicable for more than two types of transactions i.e., for multi class classification. Suppose there are three types of transactions, Genuine (G), Suspicious (S) and Fraud. Hence, the confusion matrix will become for such case as following:

Table 3.2

Confusion Matrix for multi class fraud detection system

		Predicted Value		
		Genuine (G)	**Suspicious (S)**	**Fraud (F)**
True Value	**Genuine (G)**	TG	FS	FF
	Suspicious (S)	FG	TS	FF′
	Fraud (F)	FG′	FS′	TF

Above Confusion Matrix is a 3x3 table (Table 3.2) which contains count values of True Genuine (TG), False Suspicious (FS), False Fraud (FF), False Genuine (FG), True Suspicious (TS), and True Fraud (TF). Various performance metrics can be calculated from the above confusion matrix as following:

$$\mathbf{Recall}\ (\mathbf{G}) = \frac{\mathbf{TG}}{\mathbf{TG+FS+FF}} \tag{20}$$

$$\mathbf{Recall}\ (\mathbf{S}) = \frac{\mathbf{TS}}{\mathbf{TS+FG+FF'}} \tag{21}$$

$$\mathbf{Recall}\ (\mathbf{F}) = \frac{\mathbf{TF}}{\mathbf{TF+FG'+FS'}} \tag{22}$$

$$\mathbf{G-Mean} = \sqrt[3]{\mathbf{Recall(G)} * \mathbf{Recall(S)} * \mathbf{Recall(F)}} \tag{23}$$

$$\mathbf{Accuracy}\ = \frac{\mathbf{TG+TS+TF}}{\mathbf{TG+TS+TF+FS+FF+FG+FS'+FF'+FG'}} \tag{24}$$

3.1 Research Hypothesis

The research hypothesis is also explained as below.

Null Hypothesis (H0): To predict correctly that the transaction is a fraud.

Alternate Hypothesis (H1): To predict incorrectly that the transaction is not a fraud.

Fraud Detection rate (TPR) of the system maximizes when the null hypothesis is accepted and reduces when the alternate hypothesis is accepted by rejecting the null hypothesis. Hence, the research problem for our work is to maximize the TPR so that maximum number of frauds can be detected and loss to the users and financial institutions can be minimized.

3.2 Research Objectives

The research objectives that have been undertaken during this research are:

- Preprocessing of existing raw data and feature selection to obtain relevant parameters in dataset.
- To propose a system for online transaction fraud detection using deep learning.
- To evaluate and compare the performance of the proposed system with the existing systems.

In next chapters, each objective will be explained in detail and the methods used to achieve the objective will also be discussed in detail.

Chapter-4

Data Preprocessing and Feature Selection

The first objective is to perform the preprocessing of existing raw data and feature selection to obtain relevant parameters in dataset.

4.1 Datasets

In this study, two binary datasets have been used, which contain real transactional data, and they exhibited imbalance nature where fraud transactions are less than those of genuine transactions. The description of datasets has been presented in Table 4.1.

Table 4.1

Description of Binary Datasets

Name of Dataset	**Total transactions**	**Genuine transactions**	**Fraud transactions**	**Class Imbalance Level [94]**
IEEE CIS [95]	590540	569877	20663	27.58: 1
European Credit Card [96]	284807	284315	492	577.88:1

Apart from these two binary datasets, one multi class financial dataset explained in section 4.1.3 containing three different types of transactions has also been used for validation of our proposed framework for the multi class problem.

The various techniques have been followed to preprocess all datasets and the flow chart for steps performed has been visualized in the Figure 4.1. Firstly, the duplicate rows have been checked if there are any so that the redundant rows can be removed from the dataset. After that, numerical and categorical features have been processed and missing values also have been handled in the datasets. The different techniques have been followed for different datasets which have also been explained in detail further for each dataset separately.

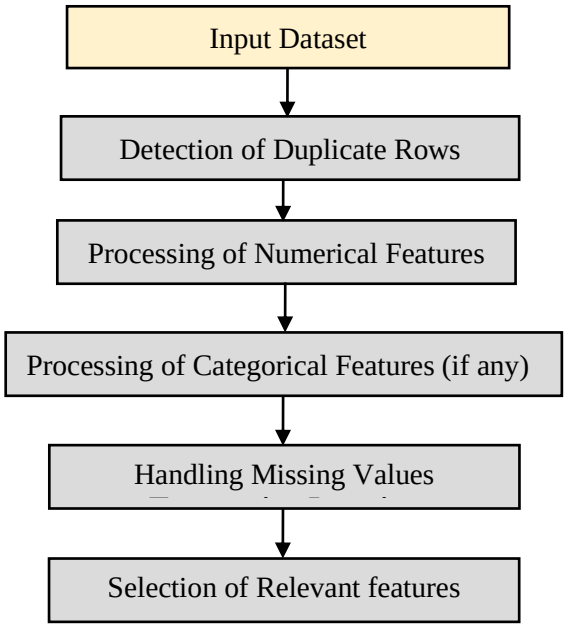

Figure 4.1: Flow Chart for steps performed for Preprocessing of datasets

4.1.1 IEEE-CIS Dataset

The dataset used is IEEE-CIS dataset available on Kaggle in which the probability of an online transaction whether it is fraudulent or not has to predict. The dataset was split into two separate tables, which contain the identity and transaction details of a transaction having one common feature TransactionID which contains unique id of a transaction. Both tables have been combined using this common feature. Snapshot of dataset is shown in Figure 4.2.

TransactionID	isFraud	TransactionDT	TransactionAmt	ProductCD	card1	card2	card3	card4	card5	card6	addr1	addr2	dist1	dist2	P_emaildomain	R_emaildomain	C1	C2	C3
2987000	0	86400	68.5	W	13926	NaN	150.0	discover	142.0	credit	315.0	87.0	19.0	NaN	NaN	NaN	1.0	1.0	0.0
2987001	0	86401	29.0	W	2755	404.0	150.0	mastercard	102.0	credit	325.0	87.0	NaN	NaN	gmail.com	NaN	1.0	1.0	0.0
2987002	0	86469	59.0	W	4663	490.0	150.0	visa	166.0	debit	330.0	87.0	287.0	NaN	outlook.com	NaN	1.0	1.0	0.0
2987003	0	86499	50.0	W	18132	567.0	150.0	mastercard	117.0	debit	476.0	87.0	NaN	NaN	yahoo.com	NaN	2.0	5.0	0.0
2987004	0	86506	50.0	H	4497	514.0	150.0	mastercard	102.0	credit	420.0	87.0	NaN	NaN	gmail.com	NaN	1.0	1.0	0.0
2987005	0	86510	49.0	W	5937	555.0	150.0	visa	226.0	debit	272.0	87.0	36.0	NaN	gmail.com	NaN	1.0	1.0	0.0
2987006	0	86522	159.0	W	12308	360.0	150.0	visa	166.0	debit	126.0	87.0	0.0	NaN	yahoo.com	NaN	1.0	1.0	0.0
2987007	0	86529	422.5	W	12695	490.0	150.0	visa	226.0	debit	325.0	87.0	NaN	NaN	mail.com	NaN	1.0	1.0	0.0
2987008	0	86535	15.0	H	2803	100.0	150.0	visa	226.0	debit	337.0	87.0	NaN	NaN	anonymous.com	NaN	1.0	1.0	0.0
2987009	0	86536	117.0	W	17399	111.0	150.0	mastercard	224.0	debit	204.0	87.0	19.0	NaN	yahoo.com	NaN	2.0	2.0	0.0
2987010	0	86549	75.887	C	16496	352.0	117.0	mastercard	134.0	credit	NaN	NaN	NaN	NaN	gmail.com	gmail.com	1.0	4.0	0.0
2987011	0	86555	16.495	C	4461	375.0	185.0	mastercard	224.0	debit	NaN	NaN	NaN	30.0	hotmail.com	hotmail.com	1.0	1.0	0.0
2987012	0	86564	50.0	W	3786	418.0	150.0	visa	226.0	debit	204.0	87.0	NaN	NaN	verizon.net	NaN	4.0	2.0	0.0
2987013	0	86585	40.0	W	12866	303.0	150.0	visa	226.0	debit	330.0	87.0	NaN	NaN	aol.com	NaN	6.0	5.0	0.0
2987014	0	86596	10.5	W	11839	490.0	150.0	visa	226.0	debit	226.0	87.0	NaN	NaN	yahoo.com	NaN	1.0	1.0	0.0
2987015	0	86618	57.95	W	7055	555.0	150.0	visa	226.0	debit	315.0	87.0	3.0	NaN	NaN	NaN	4.0	4.0	0.0
2987016	0	86620	30.0	H	1790	555.0	150.0	visa	226.0	debit	170.0	87.0	NaN	NaN	aol.com	NaN	1.0	1.0	0.0
2987017	0	86668	100.0	H	11492	111.0	150.0	mastercard	219.0	credit	204.0	87.0	NaN	NaN	yahoo.com	NaN	1.0	1.0	0.0
2987018	0	86725	47.95	W	4663	490.0	150.0	visa	166.0	debit	184.0	87.0	5.0	NaN	gmail.com	NaN	1.0	1.0	0.0
2987019	0	86730	186.0	W	7005	111.0	150.0	visa	226.0	debit	264.0	87.0	NaN	NaN	gmail.com	NaN	2.0	5.0	0.0
2987020	0	86761	39.0	W	7875	314.0	150.0	mastercard	224.0	debit	299.0	87.0	0.0	NaN	gmail.com	NaN	1.0	1.0	0.0
2987021	0	86769	159.95	W	11401	543.0	150.0	mastercard	117.0	debit	204.0	87.0	NaN	NaN	gmail.com	NaN	127.0	120.0	0.0
2987022	0	86786	50.0	H	1724	583.0	150.0	visa	226.0	credit	299.0	87.0	NaN	NaN	gmail.com	NaN	1.0	1.0	0.0
2987023	0	86808	107.95	W	2392	360.0	150.0	mastercard	166.0	debit	126.0	87.0	4.0	NaN	gmail.com	NaN	1.0	1.0	0.0
2987024	0	86821	73.95	W	10112	360.0	150.0	visa	166.0	debit	264.0	87.0	NaN	NaN	gmail.com	NaN	3.0	5.0	0.0

Figure 4.2: Snapshot of IEEE CIS dataset

Class Labelling

The target feature of the dataset is ‘isFraud’ having zero and one value for the genuine transaction and fraud transaction respectively.

Class Imbalance in the dataset

The total number of rows in the dataset is 590540. Fraud transactions are very few. Total transactions’ 3.499% are fraud transactions while rest are genuine ones.

Features Description

There are total 434 features, the description of them has been given as following:

Table 4.2

Description of features of IEEE-CIS Dataset

Feature Name	**Type**	**Feature Description**	**No of features**
TransactionID	Numerical	Transaction ID (unique)	1
TransactionDT	Numerical	Timedelta of transaction (unique)	1
isFraud	Numerical	Target Feature	1
TransactionAMT	Numerical	Transaction payment amount	1
ProductCD	Categorical	Code of the Product	1
card1 - card6	Categorical	Payment card information	6
addr1 - addr2	Categorical	Address	2
dist1 - dist2	Numerical	Distance	2
P_emaildomain and R_emaildomain	Categorical	Purchaser and recipient email domain	2
C1-C14	Numerical	Counting, such as how many addresses are found to be associated with the payment card, etc. The actual meaning is masked	14
D1-D15	Numerical	Time delta, such as days between previous transaction, etc	15
M1-M9	Categorical	Match, such as names on card and address, etc.	9
DeviceType and DeviceInfo	Categorical	Device information utlized for performing transaction	2
Identity information features (id_01 - id_38)	Numerical (id_01 - id_11) + Categorical (id_12 - id_38)	Identity information – network connection information (IP, ISP, Proxy, etc) and digital signature (UA/browser/os/version, etc) associated with transactions.	38

Vesta engineered features (V1 - V339)	Numerical	Vesta engineered rich features, including ranking, counting, and other entity relations	339

4.1.1.1 Data Preprocessing

Preprocessing of dataset has been performed before feeding it to the DNN model and the steps performed for preprocessing are as the following.

Detecting Duplicate Rows

The dataset has no duplicate transactions.

Processing numerical features

It has been observed that many of the numerical features are rightly skewed. Log transformation has been used to convert the rightly skewed continuous data into a normal distributed format. A right skewed numerical feature x has been log-transformed using the $\log_{10}(x + 1 - \min(0, \min(x))$ equation. It takes care of the missing values, zeroes and negative values present in the feature if any. For example, the TransactionAMT before log transformation and after log transformation has been shown in Figures 4.3 and 4.4. After log transformation, the skewness of feature decreased.

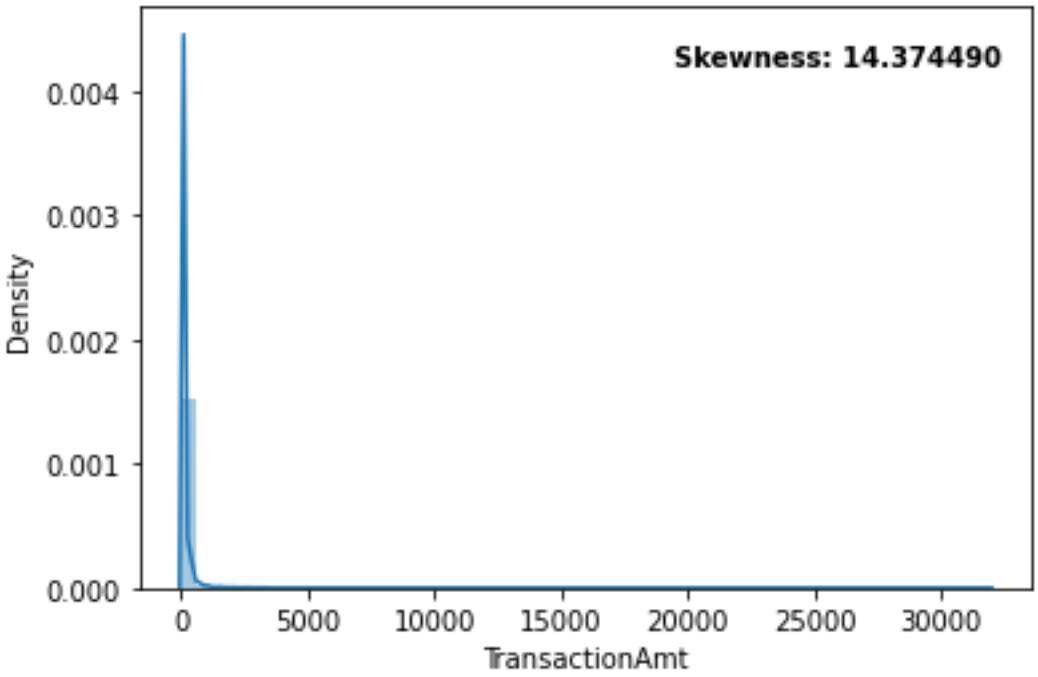

Figure 4.3: Transaction Amount before Log Transformation

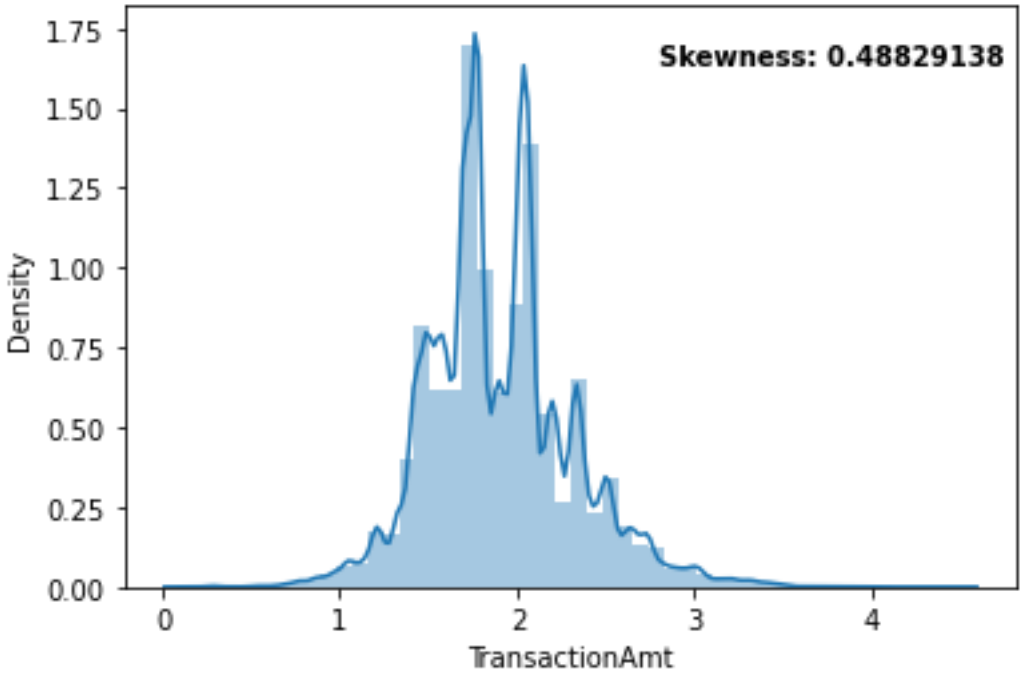

Figure 4.4: Transaction Amount after Log Transformation

Standardization of the numerical features was performed after log transformation by using the following z-score [97] which can be calculated using following formula:

$$\mathbf{z} = \frac{\mathbf{value - mean}}{\mathbf{standard\ deviation}} \tag{25}$$

After standardization, all numerical features have zero mean and standard deviation of one and all features have been mapped to same scale. The TransactionAMT feature after standardization has been shown in Figure 4.5. Also, there is no significant change in skewness of data after standardization.

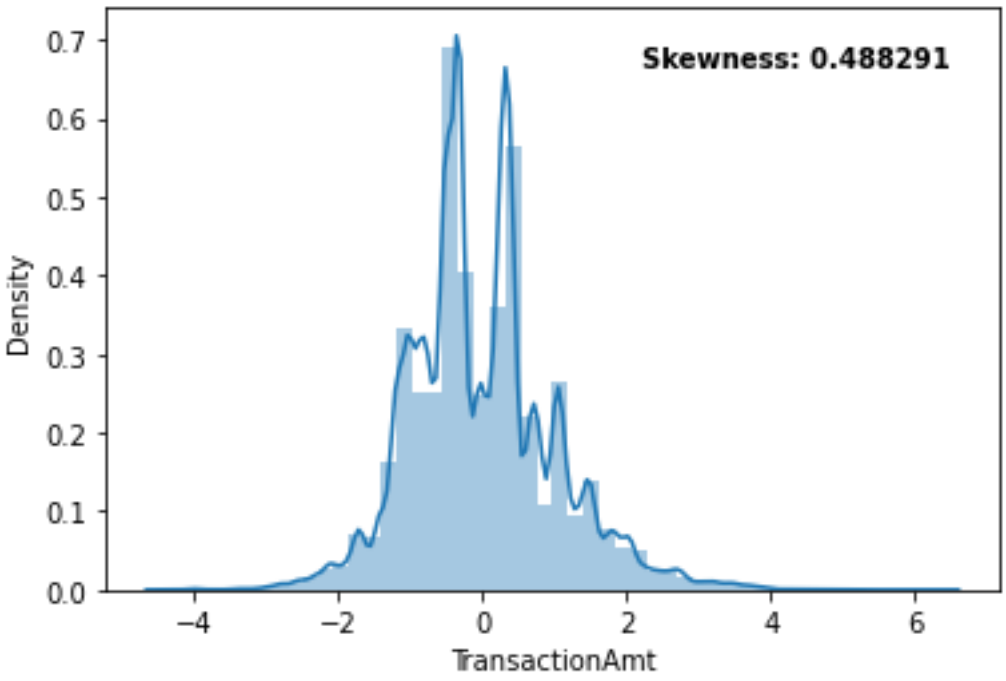

Figure 4.5: Transaction Amount after Standardization

Handling Missing Values

Handling missing values is a crucial step in preprocessing of the dataset. As most of the numerical features in the dataset contain missing values as visualized in Figure 4.6 and hence, they have been replaced with zero. Along with replacing missing values, the missing indicators have been used.

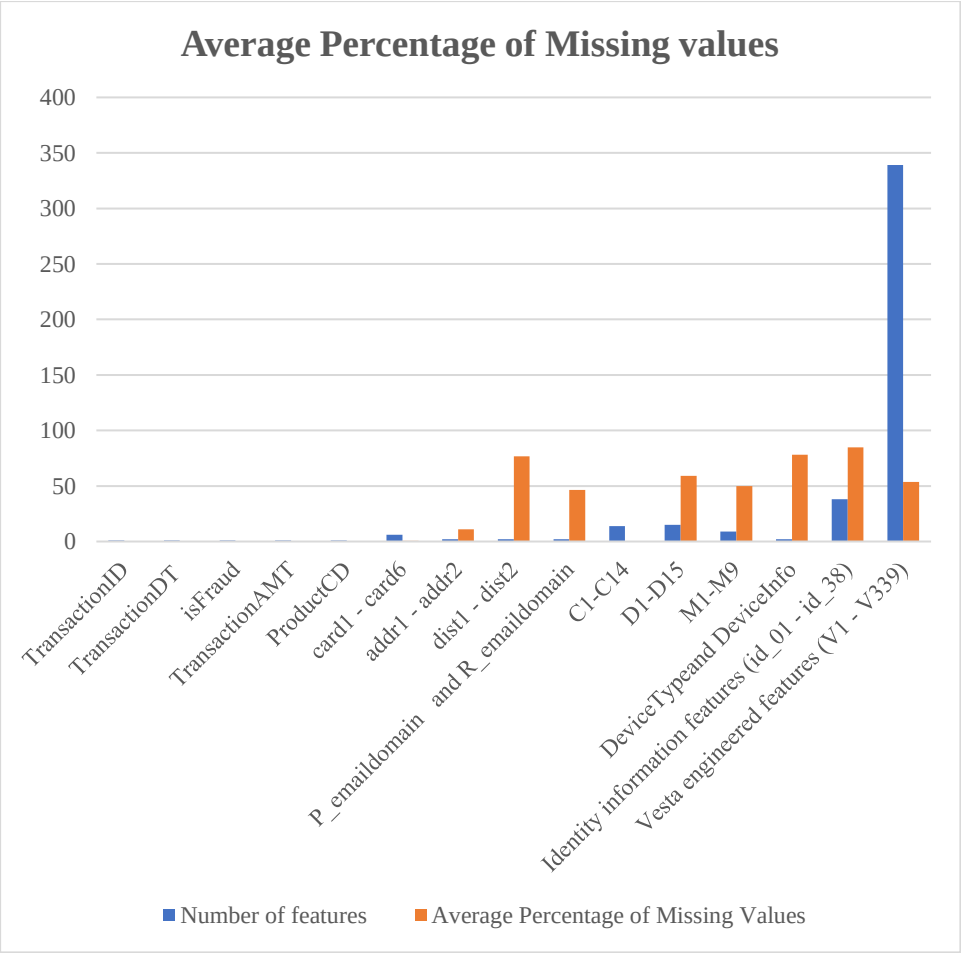

Figure 4.6: Average Percentage of Missing Values in set of features

Missing indicators [98] increase the dimensionality of the dataset. After treating missing values, the number of 32 selected numerical features after feature selection step explained ahead got increased to 49. It has been observed that this procedure of adding missing indicators has increased the model's prediction performance in terms of validation data.

Processing categorical features

The most frequent categories have been used only hence reducing the cardinality of festures and the least frequent features are renamed as ‘Other’.

The categorical features were then converted into numerical features after applying one hot encoding [99]. The resulting 509 features had a higher

dimensionality after applying one hot encoding.

Splitting into Training and Test Data

Splitting of dataset has been performed in stratified manner (80:20) for training and test data as per Table 4.3.

Table 4.3

Stratified Splitting of IEEE CIS Data

Dataset	Total Transactions	Fraudulent Transactions	Percentage of Fraudulent Transactions
Training Data	4,72,432	16,530	3.499
Test Data	1,18,108	4,133	3.499

4.1.1.2 Feature Selection

TransactionID, TransactionDT (timedelta) features are not useful as they contain unique values and have not been included. It has been found that identity information features (id_01 - id_38) and Vesta engineered features (V1 - V339) contain a lot of missing values. All of identity information features (id_01 - id_38) features and the 159 out of 339 of Vesta engineered features (V1 - V339) contain more than 75 percent missing values as shown in following Tables 4.4 and 4.5.

Table 4.4

Missing Value Percentage in Identity Features

Identity Feature	Missing Values (In Percentage)
id_24	99.196
id_25	99.131
id_07	99.127
id_08	99.127
id_21	99.126

id_26	99.126
id_27	99.125
id_23	99.125
id_22	99.125
id_18	92.361
id_03	88.769
id_04	88.769
id_33	87.589
id_10	87.312
id_09	87.312
id_30	86.865
id_32	86.862
id_34	86.825
id_14	86.446
id_13	78.440
id_16	78.098
id_05	76.824
id_06	76.824
id_20	76.418
id_19	76.408
id_17	76.400
id_31	76.245
id_02	76.145
id_29	76.127
id_11	76.127
id_28	76.127
id_37	76.126
id_36	76.126
id_15	76.126
id_35	76.126
id_38	76.126

id_01	75.576
id_12	75.576

.

Table 4.5

Missing Value Percentage in Vesta Engineered Features

Vesta Engineered Feature	**Missing Values (In Percentage)**
V142	86.124
V158	86.124
V140	86.124
V162	86.124
V141	86.124
V161	86.124
V157	86.124
V146	86.124
V156	86.124
V155	86.124
V154	86.124
V153	86.124
V149	86.124
V147	86.124
V148	86.124
V163	86.124
V139	86.124
V138	86.124
V160	86.123
V151	86.123
V152	86.123
V145	86.123
V144	86.123

V143	86.123
V159	86.123
V164	86.123
V165	86.123
V166	86.123
V150	86.123
V337	86.055
V333	86.055
V336	86.055
V335	86.055
V334	86.055
V338	86.055
V339	86.055
V324	86.055
V332	86.055
V325	86.055
V330	86.055
V329	86.055
V328	86.055
V327	86.055
V326	86.055
V322	86.055
V323	86.055
V331	86.055
V278	77.913
V277	77.913
V252	77.913
V253	77.913
V254	77.913
V257	77.913
V258	77.913

V242	77.913
V261	77.913
V262	77.913
V263	77.913
V264	77.913
V249	77.913
V266	77.913
V267	77.913
V268	77.913
V269	77.913
V273	77.913
V274	77.913
V275	77.913
V276	77.913
V265	77.913
V260	77.913
V247	77.913
V246	77.913
V240	77.913
V237	77.913
V236	77.913
V235	77.913
V233	77.913
V232	77.913
V231	77.913
V230	77.913
V229	77.913
V228	77.913
V226	77.913
V225	77.913
V224	77.913

V223	77.913
V219	77.913
V218	77.913
V217	77.913
V243	77.913
V244	77.913
V248	77.913
V241	77.913
V212	76.355
V211	76.355
V214	76.355
V213	76.355
V196	76.355
V205	76.355
V183	76.355
V216	76.355
V206	76.355
V186	76.355
V187	76.355
V192	76.355
V207	76.355
V215	76.355
V181	76.355
V182	76.355
V191	76.355
V167	76.355
V168	76.355
V199	76.355
V193	76.355
V172	76.355
V173	76.355

V202	76.355
V203	76.355
V176	76.355
V177	76.355
V178	76.355
V179	76.355
V204	76.355
V190	76.355
V194	76.324
V200	76.324
V189	76.324
V188	76.324
V185	76.324
V184	76.324
V180	76.324
V175	76.324
V174	76.324
V171	76.324
V170	76.324
V169	76.324
V195	76.324
V201	76.324
V197	76.324
V198	76.324
V209	76.324
V208	76.324
V210	76.324
V245	76.053
V271	76.053
V234	76.053
V222	76.053

V238	76.053
V239	76.053
V227	76.053
V250	76.053
V272	76.053
V270	76.053
V251	76.053
V220	76.053
V255	76.053
V256	76.053
V259	76.053
V221	76.053
V3	47.293
V9	47.293
V5	47.293
V11	47.293
V10	47.293
V8	47.293
V7	47.293
V6	47.293
V4	47.293
V2	47.293
V1	47.293
V35	28.613
V40	28.613
V41	28.613
V39	28.613
V38	28.613
V51	28.613
V37	28.613
V52	28.613

V36	28.613
V50	28.613
V48	28.613
V42	28.613
V43	28.613
V44	28.613
V46	28.613
V47	28.613
V45	28.613
V49	28.613
V80	15.099
V87	15.099
V88	15.099
V89	15.099
V90	15.099
V91	15.099
V92	15.099
V93	15.099
V94	15.099
V86	15.099
V79	15.099
V85	15.099
V75	15.099
V84	15.099
V77	15.099
V83	15.099
V78	15.099
V82	15.099
V81	15.099
V76	15.099
V72	13.055

V74	13.055
V73	13.055
V71	13.055
V65	13.055
V68	13.055
V58	13.055
V70	13.055
V53	13.055
V54	13.055
V55	13.055
V56	13.055
V57	13.055
V59	13.055
V67	13.055
V60	13.055
V61	13.055
V62	13.055
V63	13.055
V64	13.055
V66	13.055
V69	13.055
V21	12.882
V22	12.882
V23	12.882
V34	12.882
V33	12.882
V32	12.882
V31	12.882
V30	12.882
V29	12.882
V28	12.882

V27	12.882
V25	12.882
V24	12.882
V26	12.882
V16	12.882
V15	12.882
V20	12.882
V14	12.882
V19	12.882
V18	12.882
V17	12.882
V12	12.882
V13	12.882
V296	0.215
V289	0.215
V288	0.215
V283	0.215
V282	0.215
V281	0.215
V300	0.215
V301	0.215
V313	0.215
V314	0.215
V315	0.215
V104	0.053
V109	0.053
V110	0.053
V111	0.053
V112	0.053
V106	0.053
V105	0.053

V102	0.053
V103	0.053
V96	0.053
V101	0.053
V100	0.053
V99	0.053
V98	0.053
V97	0.053
V95	0.053
V135	0.053
V134	0.053
V107	0.053
V133	0.053
V132	0.053
V131	0.053
V130	0.053
V129	0.053
V128	0.053
V127	0.053
V126	0.053
V125	0.053
V124	0.053
V123	0.053
V122	0.053
V121	0.053
V120	0.053
V119	0.053
V118	0.053
V117	0.053
V116	0.053
V115	0.053

V114	0.053
V113	0.053
V136	0.053
V137	0.053
V108	0.053
V311	0.002
V321	0.002
V294	0.002
V306	0.002
V305	0.002
V304	0.002
V303	0.002
V302	0.002
V299	0.002
V298	0.002
V297	0.002
V295	0.002
V293	0.002
V308	0.002
V292	0.002
V291	0.002
V290	0.002
V287	0.002
V286	0.002
V285	0.002
V284	0.002
V280	0.002
V279	0.002
V320	0.002
V307	0.002
V309	0.002

V312	0.002
V316	0.002
V317	0.002
V318	0.002
V319	0.002
V310	0.002

Thus, to check whether these sets of features are useful or not, we have used the Backward feature elimination (BFE) method [100] which is a wrapper-style feature selection method. The BFE algorithm starts by selecting the complete set of features and removes one or a set of features at a time if it gets better or doesn't change the performance. If the prediction performance of the learning model decreases after the removal of features, then those features are added back to the list. The same procedure has been adopted in this work.

For validation purposes, we selected 20% validation data from training data in a stratified manner. Firstly, we trained our deep learning model using all features and monitored the TPR value of the validation data. Then, the model was trained by excluding identity information features (id_01 - id_38) only and the impact on the performance of the model in terms of TPR of validation data was checked which was none, and hence these features have been excluded. After that, the deep learning model was trained by excluding Vesta engineered features (V1 - V339) only, and the performance criterion set i.e., TPR value was not much impacted after removing these features and hence these features have also been excluded. The flow chart of the backward elimination method adopted has been shown in Figure 4.7.

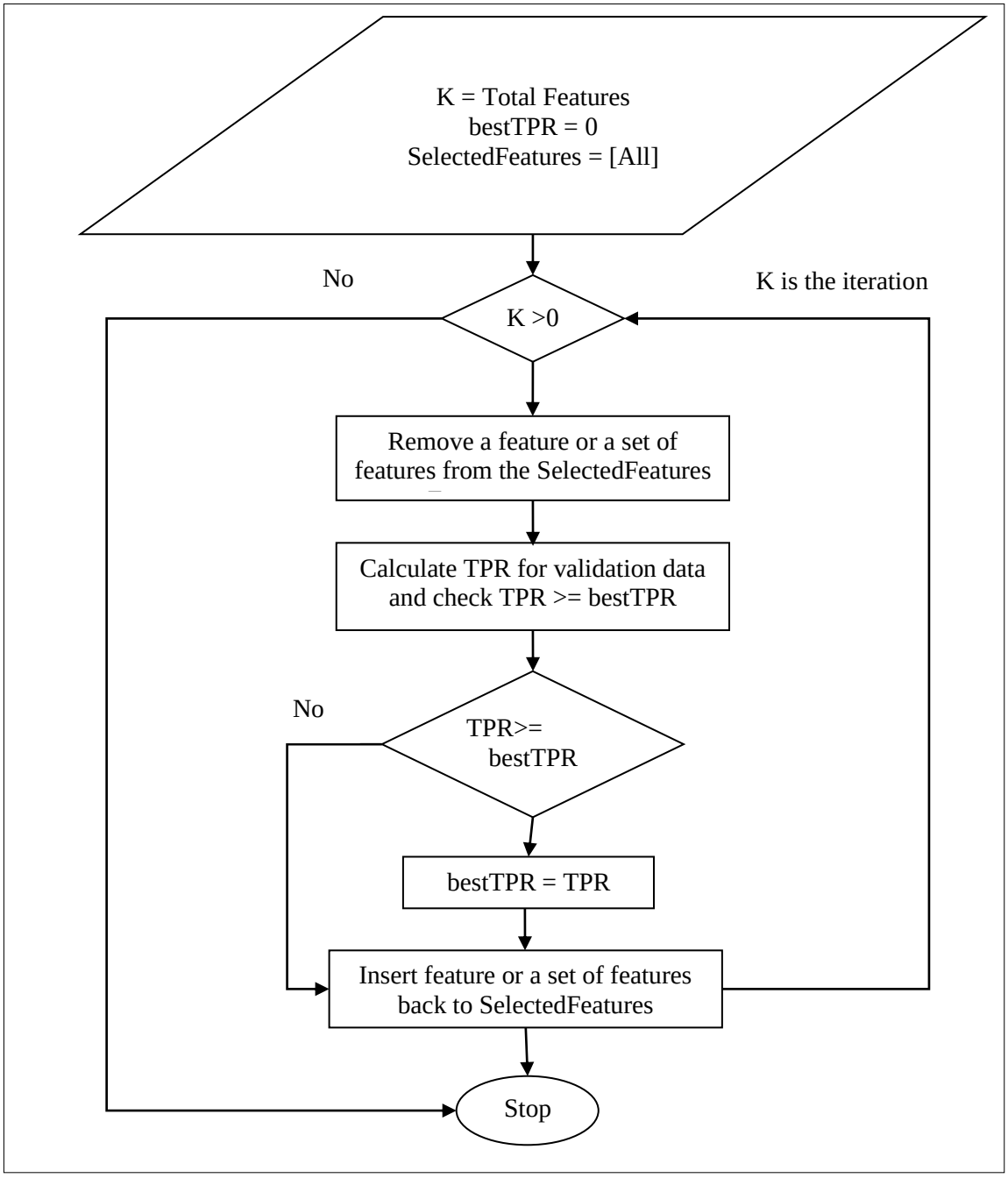

Figure 4.7: Backward Feature Elimination (BFE) method for feature selection

Out of the 54 selected features, numerical features are 32, while the remaining are categorical features as the model's performance in terms of TPR of validation data and computational speed also did not improve when these identity information and vesta engineered features were included. Also, after removing these two sets of features, we have further enhanced the model's performance by adding missing

indicators when missing values were handled in 32 numerical features as explained in data preprocessing.

4.1.2 European Credit Card Dataset

This dataset contains the total number of credit card transactions made by European consumers in September 2013. Out of these 284,807 transactions, only 492 were fraudulent. The snapshot of data has been shown in Figure 4.8.

Time	V1	V2	V3	V4	V5	V6	V7	V8	V9
0.0	-1.3598071336738	-0.0727811733098497	2.536346737796914	1.37815522427443	-0.338320769942518	0.462387777762292	0.239598554061257	0.0986979012610507	0.363786969611213
0.0	1.19185711131486	0.26615071205963	0.16648011335321	0.448154078460911	0.0600176492822243	-0.0823608088155687	-0.0788029833323113	0.0851016549148104	-0.255425128109186
1.0	-1.35835406159823	-1.34016307473609	1.77320934263119	0.379779593034328	-0.503198133318193	1.80049938079263	0.791460956450422	0.247675786588991	-1.51465432260583
1.0	-0.966271711572087	-0.185226008082898	1.79299333957872	-0.863291275036453	-0.0103088796030823	1.24720316752486	0.23760893977178	0.377435874652262	-1.38702406270197
2.0	-1.15823309349523	0.877736754848451	1.548717846511	0.403033933955121	-0.407193377311653	0.095921462468425 6	0.592940745385545	-0.270532677192282	0.817739308235294
2.0	-0.425965884412454	0.960523044882985	1.14110934232219	-0.168252079760302	0.42098688077219	-0.0297275516639742	0.476200948720027	0.260314333074874	-0.56867137571251
4.0	1.22965763450793	0.141003507049326	0.0453707735899449	1.20261273673594	0.191880988597645	0.272708122899098	-0.0051590028825098	0.0812129398830894	0.464959994783886
7.0	-0.644269442348146	1.41796354547385	1.0743803763556	-0.492199018495015	0.948934094764157	0.428118462833089	1.12063135838353	-3.80786423873589	0.615374730667027
7.0	-0.894286082202 82	0.286157196276544	-0.113192212729871	-0.271526130088604	2.66959986595986	3.72181806112751	0.370145127676916	0.851084443200905	-0.392047586798604
9.0	-0.338261752425 75	1.11959337641566	1.04436655157316	-0.222187276738296	0.499360806497 27	-0.246761100619 91	0.651583206489972	0.0695385865186387	-0.736727316364109
10.0	1.44904378114715	-1.17633882535966	0.913859832832795	-1.37566665499943	-1.97138316545323	-0.629152138897 34	-1.4232356010359	0.0484558879088564	-1.72040839292037
10.0	0.384978215180 95	0.616109459176472	-0.874299702595052	-0.0940186259679115	2.924584378388 17	3.317027166826156	0.470454671805879	0.538247228376 95	-0.558894612428441
10.0	1.249998742053	-1.22163680921816	0.383930151282291	-1.23489868766892	-1.48541947377961	-0.753230164566149	-0.689404975426345	-0.227487227519552	-2.09401057344842
11.0	1.06937358788 19	0.287722129331455	0.828612726634281	2.71252042961718	-0.178398016248009	0.337543730282968	-0.0967168617395962	0.115981735546597	-0.221082566236194
12.0	-2.7918547659339	-0.327770756658658	1.64175016056605	1.76747274389883	-0.136588446465306	0.807596468265 32	-0.422911389711497	-1.90710747624096	0.755712908314791
12.0	-0.752417042956605	0.345485415344747	2.05732291276727	-1.46864329840046	-1.1583936804082	-0.0778498291166733	-0.608581418236123	0.0036034843620184	-0.436166983515744
12.0	1.10321543528383	-0.0402962145973447	1.2673320885949	1.28909146962552	-0.735997163604068	0.288069162976262	-0.586056786337461	0.189379713679593	0.782332891785191
13.0	-0.436905071360625	0.918966212909322	0.924590774388 17	-0.727219053596792	0.915678718106307	-0.127867352079254	0.707641607333935	0.0879623554672504	-0.665271354133 64
14.0	-5.401257666315825	-5.45014783420644	1.18630463143652	1.73623880012095	3.04910587764025	-1.76340557365201	-1.55937369907953	0.160841747266769	1.233089740418 88
15.0	1.4929359769862	-1.02934573189487	0.45479473374366	-1.43802587991702	-1.55543410136344	-0.720961147043557	-1.08066413038614	-0.0531271179483221	-1.9786815953872

Figure 4.8: Snapshot of European Credit Card Dataset

Features Description

It contains 31 numerical input variables which are the fraud detection parameters. It only includes numerical input variables resulting from a PCA transformation. The original features and background information about the data has not been provided because of privacy concerns. Characteristics V1, V2, ..., V28 are the main components obtained with PCA, only ‘Time' and 'Amount' are the features not transformed with PCA. The 'Time' attribute includes the seconds between each transaction and the dataset's first transaction. The 'Amount' feature is the transaction’s amount and 'Class' feature is the response factor and in the event of fraud, it takes value 1 and 0 otherwise.

4.1.2.1 Data Preprocessing

Detecting Duplicate Rows

Duplicate rows have been checked in the dataset. It has been found that total 1081 transactions are duplicate in the dataset, out of which 91 are fraud transactions and 1062 are genuine transactions. All duplicate rows have been deleted. After deletion of duplicate rows, rows present in the dataset are as per Table 4.6.

Table 4.6

Transactions in Credit Card Dataset after removal of duplicate rows

Total Transactions	283726
Genuine Transactions	283253
Fraud Transactions	473

Detecting Missing Values

Missing values have not been found in any of the features.

Table 4.7

Missing Percentage in features of Credit Card Dataset

Feature	**Missing Percentage**
'Time'	0.0
'V16'	0.0
'Amount'	0.0
'V28'	0.0
'V27'	0.0
'V26'	0.0
'V25'	0.0
'V24'	0.0
'V23'	0.0
'V22'	0.0

‘V21’	0.0
‘V20’	0.0
‘V19’	0.0
‘V18’	0.0
‘V17’	0.0
‘V15’	0.0
‘V1’	0.0
‘V14’	0.0
‘V13’	0.0
‘V12’	0.0
‘V1’	0.0
‘V10’	0.0
‘V9’	0.0
‘V8’	0.0
‘V7’	0.0
‘V6’	0.0
‘V5’	0.0
‘V4’	0.0
‘V3’	0.0
‘V2’	0.0
‘Class’	0.0

Feature Scaling

The datasets have 31 columns, and out of which 28 are the principal components generated after the features have been transformed using the algorithm Principal Component Analysis (PCA). Three columns ‘Amount’, ‘Time’ and ‘Class’ have not been transformed. Class column shows the transactions that are genuine or fraudulent.

Robust Scaling Method of sklearn library [101] in Python has been used for performing for scaling the features ‘Time’ and ‘Amount’ since there are outliers present in the amount feature as shown in Figure 4.9 and this method uses

interquartile range to scale the data. The genuine transactions have a large mean value and many outliers, on the other hand fraudulent transactions have a small mean value.

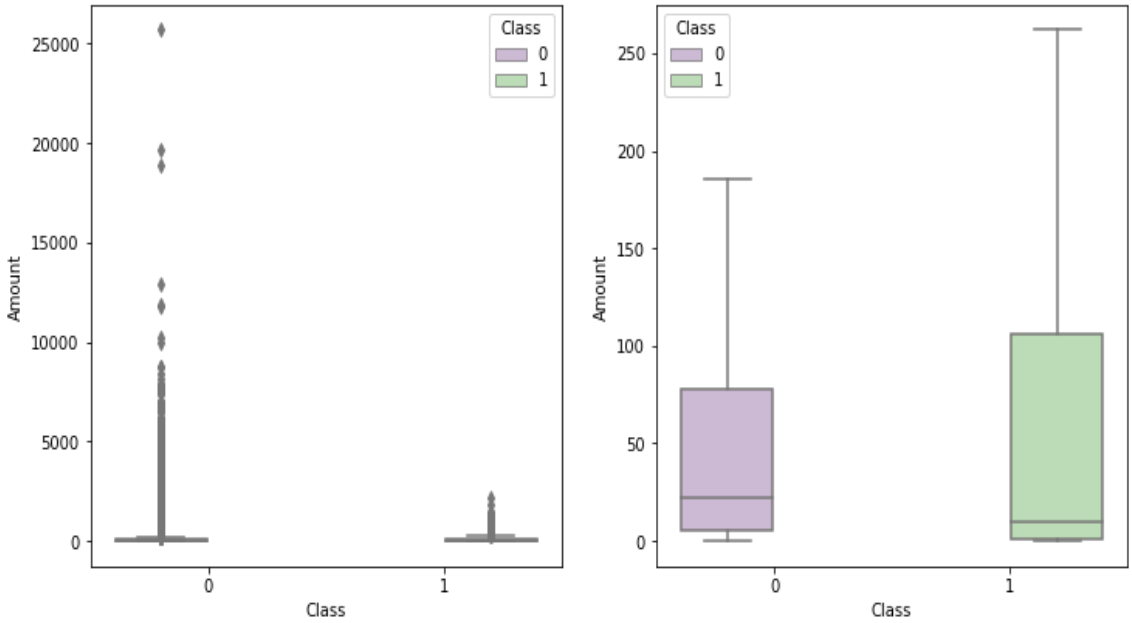

Figure 4.9: Boxplot for Amount vs Class

After preprocessing the dataset was split in two parts: training data and test data in stratified manner so that ratio of genuine to fraud transactions remains same in both datasets as shown in Table 4.8.

Table 4.8

Stratified Splitting of European Data

Dataset	**Total Transactions**	**Fraudulent Transactions**	**Percentage of Fraudulent Transactions**
Training Data	2,26,980	378	0.167
Test Data	56,746	95	0.167

4.1.2.2 Feature Selection

Correlation coefficient [102] has been used to find the correlation among features, hence a correlation matrix has been plotted for finding the correlation between all two set of features.

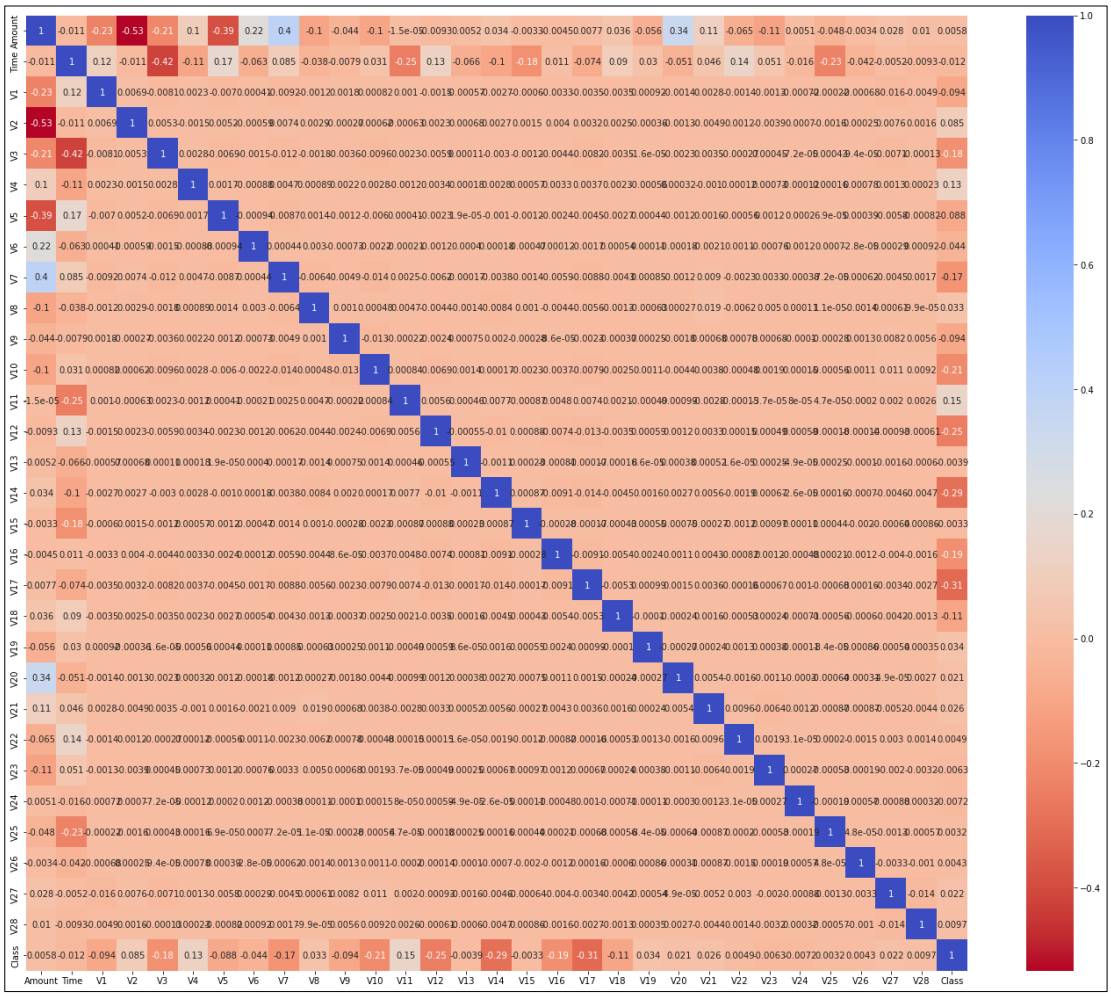

Figure 4.10: Correlation Matrix Plot for all features of Credit Card Dataset

As shown in Figure 4.10, there is no notable correlation between features except 'V2' and 'Amount' which are moderately correlated as the correlation between them is 0.53. The correlation between feature 'V2' and target feature i.e., 'Class' is less and the correlation between feature 'Amount' and 'Class' is also less. So, the decision for selecting one feature between V2 and Amount or excluding them both cannot be made. So, we also applied the backward feature elimination method (BFE) as discussed in previous section and shown in Figure 4.7 for selection of features in

this dataset. We first trained the dataset on all features and then by excluding V2 first and monitored TPR value of validation data (20% selected from training data in a stratified manner). Then by excluding only Amount feature, TPR was monitored. The performance was also checked by excluding both features. It has been found that when all features were used, then TPR achieved is maximum. Hence, all features have seemed to be important and have been used.

4.1.3 Multi Class Financial Dataset

To validate our proposed fraud detection system, a multi class dataset has also been used. Since the real datasets available are generally two class, hence due to non-availability of multi class real datasets, a synthetic financial data like a real-world data used in the finance and cost control modules of the SAP-ERP system has been used.

This third dataset is a synthetic version of financial transactions used by the researchers for anomaly detection [103] and is also available on GitHub [104]. The dataset contains 2 numerical and 7 categorical features. The target feature 'Label' has been used to categorize the transaction whether it is a regular transaction or a fraud (local or global). Hence, Label feature contains three labels 'regular', ''local' and 'global'. The snapshot of the dataset is shown in Figure 4.11. The description of all features of the data set is given as per Table 4.9.

	BELNR	WAERS	BUKRS	KTOSL	PRCTR	BSCHL	HKONT	DMBTR	WRBTR	label
0	288203	C3	C31	C9	C92	A3	B1	280979.60	0.00	regular
1	324441	C1	C18	C7	C76	A1	B2	129856.53	243343.00	regular
2	133537	C1	C19	C2	C20	A1	B3	957463.97	3183838.41	regular
3	331521	C4	C48	C9	C95	A2	B1	2681709.51	28778.00	regular
4	375333	C5	C58	C1	C19	A3	B1	910514.49	346.00	regular
5	327203	C1	C15	C6	C68	A1	B2	357627.56	704520.00	regular
6	292545	C4	C47	C2	C28	A2	B3	955576.84	128328.00	regular
7	335839	C1	C19	C1	C17	A1	B1	41769.26	0.00	regular

Figure 4.11: Snapshot of Multi-class financial dataset.

Table 4.9

Description of features

Feature	Description	Type
BELNR	Accounting document number	Numerical
WAERS	Company Code	Categorical
BUKRS	Posting Key	Categorical
KTOSL	Posted Journal Ledger Account	Categorical
PRCTR	Posted Profit Center	Categorical
BSCHL	Currency Key	Categorical
HKONT	General Ledger Account Key	Categorical
DMBTR	Amount in Local Currency	Categorical
WRBTR	Amount in Document Currency	Numerical
Label	Target Feature	Categorical

4.1.3.1 Data Preprocessing

Detecting Duplicate Rows

The dataset has no duplicate transactions.

Processing numerical features

There are only two numerical features i.e., BELNR and WRBTR. The standardization has been performed on these two features as after standardization, both numerical features have zero mean and standard deviation of one and both have been mapped to same scale.

Processing categorical features

The seven categorical features except ‘Label’ have been applied one hot encoding to transform them into numerical form. Label is the target feature used to explain true nature of transaction and contains three different labels for three classes which have further numerically encoded has 0, 1, and 2 for ‘Global’, ‘Local’ and ‘Regular’ where ‘Global’ and ‘Local’ are fraud transactions and ‘Regular’ are genuine transactions. The feature wise count of categories has been given in Table 4.10. Thus, after

applying one hot encoding on these six categorical features, their total features count becomes 616.

Table 4.10

Category wise count of Features

Feature	Count of Different Categories
WAERS	76
BUKRS	158
KTOSL	79
PRCTR	157
BSCHL	73
HKONT	73

Detecting missing values

Missing values have not been found in any of the features.

After preprocessing, the dataset was split in two parts: training data and test data in stratified manner so that ratio of genuine to fraud transactions remains same in both datasets as shown in Table 4.11.

Table 4.11

Stratified Splitting of Multi-Class Financial Data

Dataset	Total Transactions	Regular Transactions	Global Fraud Transactions	Local Fraud Transactions	% of Global Frauds	% of Local Frauds
Training Data	426407	426327	56	24	0.0131	0.0056
Test Data	106602	106582	14	6	0.0131	0.0056

4.1.3.2 Feature Selection

Backward feature elimination method (BFE) has been used for selection of relevant features in this dataset. We first trained the dataset on all features and monitored the performance of model on validation data (20% selected from training data in a stratified manner). Then by excluding one feature at a time, validation data performance was monitored. It has been observed that when all features were used, then recall values achieved for all three classes were maximum for validation data. Hence, all features have seemed to be important and have been used.

Chapter-5

Proposed Fraud Detection System

The second objective is to propose a system for fraud detection in online transactions by utilization of deep learning. Hence, a system has been proposed which detects frauds efficiently from imbalanced datasets without the need of modifying the datasets by applying algorithm-level techniques. The results generated using the proposed system have also compared with data-level and hybrid methods using deep neural network (DNN) showing the efficiency of our system which will be discussed in Chapter 7.

5.1 Handling Class Imbalance

Class imbalance in two real binary datasets have has been handled using data-level, algorithm-level, and hybrid methods. Random undersampling (RUS) and Random oversampling (ROS) have been used in data-level methods separately on both binary datasets to balance them. In RUS method, the undersampling of genuine transactions has been performed to make the frequency of both types of transactions equal. In ROS method, oversampling on fraud transactions has been performed to make the ratio of fraud transactions equal to the genuine transactions in the dataset. Since after RUS and ROS, the frequency of genuine and fraud transaction became equal, the threshold i.e., 0.5 has been used to obtain the results by utilizing cross entropy loss (CEL) function with DNN.

Threshold has been optimized in algorithm-level and hybrid methods as the data is still not completely balanced. In algorithm-level methods, data has not been modified. Various loss functions have been utilized which further combined with thresholding to automatically optimizing the threshold. The detailed explanation has been given in further section.

The hybrid method has combined the two methods i.e., RUS and ROS. In this research work, 1 % ROS was performed on fraudulent transactions and 5% RUS on

genuine transactions. The results of these experiments were then analyzed and optimized using the thresholding method. For hybrid method, CEL has been used. The flow chart (Figure 5.1) for this study shows the complete research procedure followed for binary datasets. For multi-class dataset, only algorithm-level approach has been used without modifying the data.

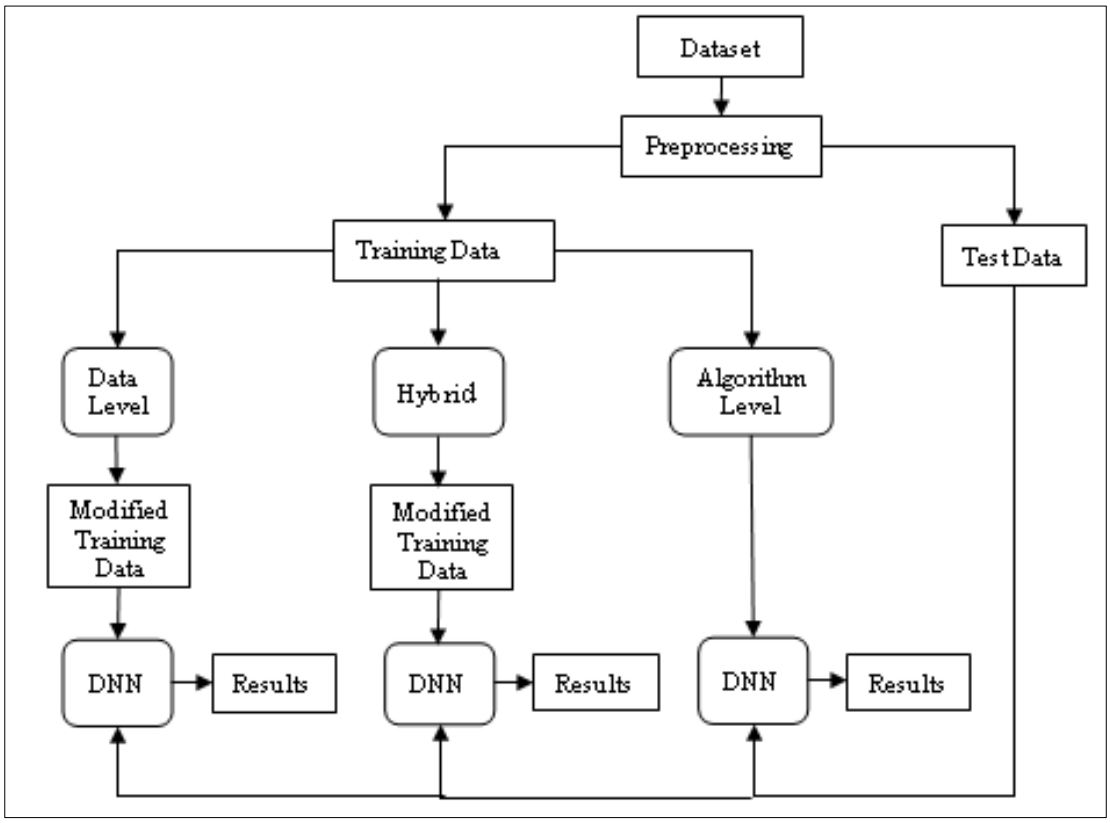

Figure 5.1: Flow Chart for Research Procedure

5.2 Proposed Fraud Detection System using Algorithm-Level Method

In algorithm-level methods, training data was used in its original form after preprocessing and feature selection. Thus, no modification of training data was done to make it balance either by under sampling or oversampling. Since the data is imbalanced hence the decision threshold was also optimized using the repeated Stratified K-fold cross validation. The optimal threshold was then used to evaluate the model's performance in terms of predicting the results of the test data. The procedure has been followed by the various loss functions as mentioned previously in Chapter 1.

Thresholding in case of binary data

As the training data is imbalanced during training of the DNN model using algorithm-level/hybrid method, thresholding has been used. Repeated Stratified K-fold cross validation with k=5 and repeat=2 been used. Thus, total folds used were 5*2 =10. Optimal thresholds were obtained using the Closest to (0,1) criteria for each 10 folds. Early stopping was used to monitor the value of TPR, and TNR and optimal threshold was obtained where TPR and TNR is maximum for each fold. The optimal threshold was used to obtain the final test results by training the DNN model from scratch using the whole training data for the same number of epochs for which the optimal threshold was obtained using the validation data. This procedure was repeated for all fold results and best test data results with maximum TPR and then, TNR were saved.

Random search procedure was then used to identify the models that were most suitable for the three datasets. The validation data was then used to evaluate the model's performance and select the best hyperparameters. The selected model's architecture then decided upon will be used to evaluate the test data. This ensures that the model's tuning does not get influenced by the test data.

DNN models with 2 hidden layers have been used for both datasets to extract features automatically. Since the dropout has been used, a large size of the network has been used [105] i.e., a large number of neurons in the hidden layers have been used. Also, if N is the number of neurons used in the first hidden layer, then N/2 number of neurons have been used in the second hidden layer and then single neuron in the output layer since its binary classification task for fraud detection as shown in Figure 5.2. Hence, a pyramid-shaped deep neural network structure has been selected which performed best during finalizing the baseline architecture of DNN. The baseline architecture of DNN is the same for both datasets. If there are more than two types of transactions i.e., in multi class problem, then neurons equal to the number of classes will be used in the output layer as shown in Figure 5.3 where three neurons have been used in output layer for three class problem.

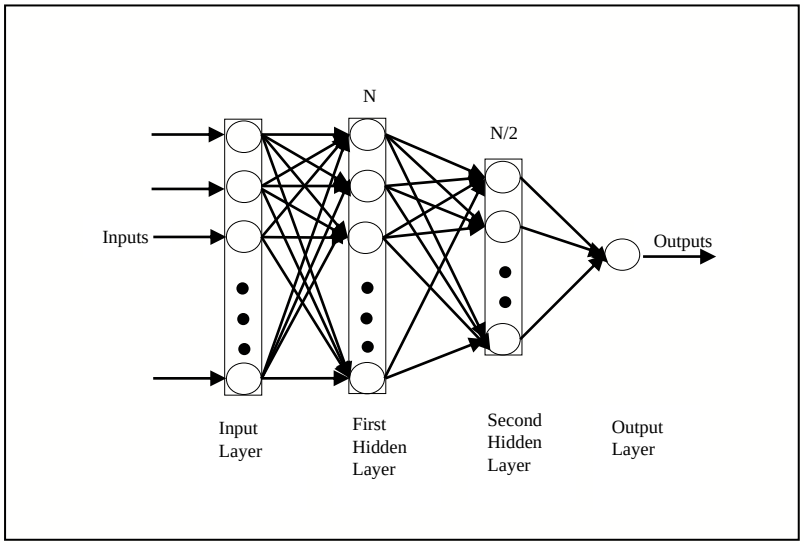

Figure 5.2: Architecture of DNN for binary classification

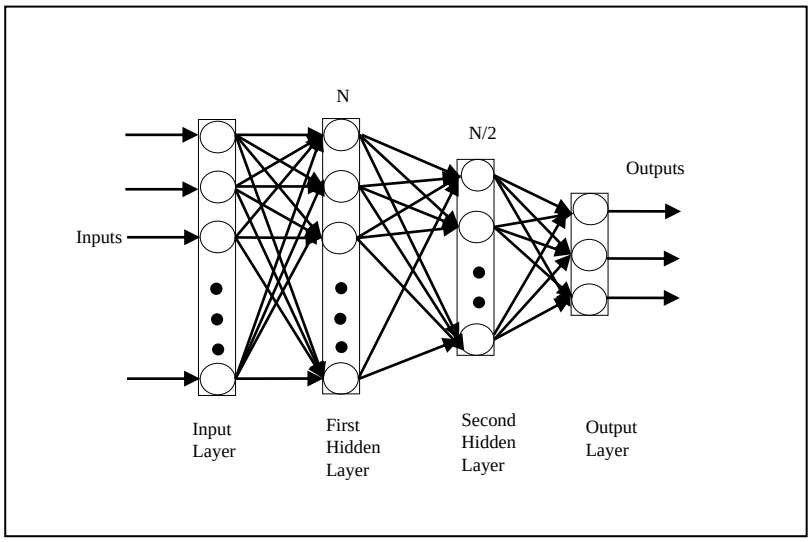

Figure 5.3: Architecture of DNN for multi class classification

Table 5.1 describes the architecture of models used for both binary class datasets. DNN models with 2 hidden layers have been used for all both datasets to extract features automatically.

Table 5.1

Architectures of Deep Neural Network (DNN) for both binary datasets

	IEEE CIS	**European Credit Card**
Layer Type	**Number of neurons**	
Input Layer	558	30
Dense Layer	512	32
Batch Normalization		
ReLU		
Dropout Rate = 0.3		
Dense Layer	256	16
Batch Normalization		
ReLU		
Dropout Rate = 0.3		
Dense Layer	1	1
Sigmoid [106]		

The batch normalization was also used to normalize the batch of inputs to the hidden layers [107]. The ReLU activation function with he_uniform weight initialization [108] was used in the hidden layers due to its good performance. Batch size of 256 transactions was used. Adam optimizer [109] was used with slow learning rate i.e., 0.0001. Since the datasets used contain only two types of transactions i.e., genuine and fraud transactions, hence single neuron and Sigmoid function has been used in the output layer. The third dataset i.e., multi class financial dataset contain more than two types of transactions, hence 3 neurons have been used in output layer along with the SoftMax function [110] as shown in Table 5.2.

Table 5.2

Architectures of Deep Neural Network (DNN) for multi-class financial dataset

	Multi Class Financial Dataset
Layer Type	**Number of neurons**
Input Layer	618
Dense Layer	256
Batch Normalization	
ReLU Activation Function	
Dropout (0.3)	
Dense Layer	128
Batch Normalization	
ReLU Activation Function	
Dropout (0.3)	
Dense Layer	3
Softmax Function	

Hence, for binary classification, Sigmoid function has been used whereas for multi-class classification, SoftMax has been used in the output layer. The difference between the two is that for Softmax, the sum of probabilities needs to be 1 whereas for Sigmoid, this is not the case as shown in figure 5.4. In case of Sigmoid activation function, the output label will be predicted as class having label 1 in binary classification if the probability is greater than or equal to the specific threshold value otherwise it will be predicted as class having label 0. On the other hand, in case of Softmax, the output label will be predicted as the class label which has got maximum probability value.

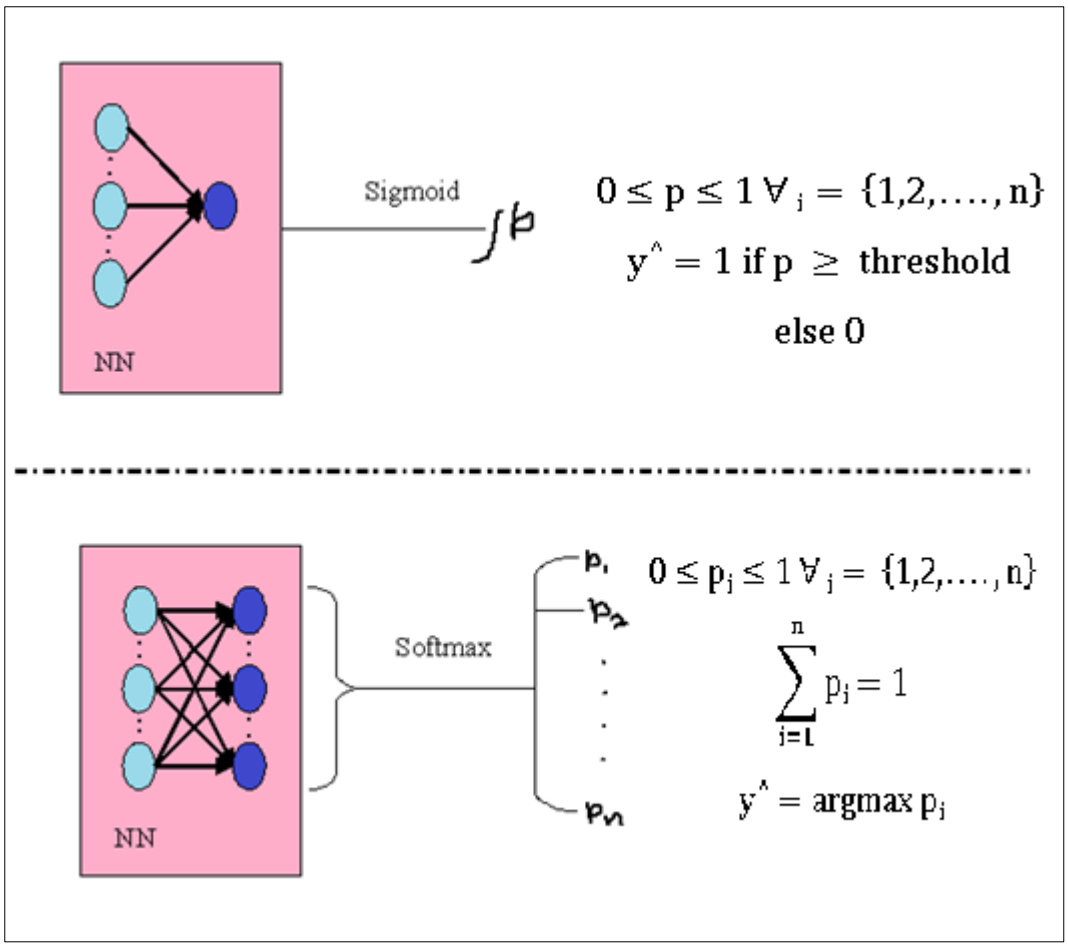

Figure 5.4: Sigmoid vs Softmax function

Initialization of Deep Neural Network (DNN)

T. Lin et al. [49] have used the prior probability of $(p = 0.01)$ detecting an object in the last layer of the model which resulted in significant improvement in results. In the last layer of the model, the output bias weights were also adjusted to prevent the majority class from significantly contributing to the value of loss during early training of the model. We have initialized bias for all experiments performed using all loss functions under study. By initializing DNN model, it has been observed that the average training time was decreased in terms of epochs.

Chapter-6

Comparison of Proposed System with Existing Systems

The third objective is to compare the proposed system's performance with that of the existing systems. First, we have compared our deep learning-based fraud detection system for binary datasets containing two types of transactions i.e., genuine and fraud and then for multi-class dataset for multi-classification problem.

6.1 Binary Classification

Our primary datasets are first two real datasets which contain only two types of transactions. So, we have done the comparison of our proposed system using these two datasets with existing systems.

The proposed system has been compared to existing machine learning models which include the K Nearest Neighbor (KNN), the Decision Tree (DT), Logistic Regression (LR), Light Gradient Boosting Machine (LGBM) and Random Forest (RF). Various hyperparameters have been tuned for machine learning models by utilizing 20% validation data obtained from training data. The number of nearest neighbors in KNN, Penalty and inverse of regularization strength for LR, maximum depth of the tree for DT, number of trees for RF and maximum depth, number of leaves in single decision tree in LGBM etc. have been tuned.

Various Loss functions utilized with Deep Neural Network have also been implemented with LGBM for comparison.

As our research work is mainly focused on detecting frauds from imbalanced

data hence, we have compared the performance of our system using an algorithm-level method. Thus, all machine learning algorithms have also been trained on imbalanced data.

DNN model trained using the Cross-Entropy loss (CEL) function has been chosen as a baseline for comparison of all algorithms. The whole procedure adopted for comparison of the proposed system with existing for binary datasets has been shown in Figure 6.1. In case of multi-class dataset, the decision threshold has not been optimized as done in binary datasets since the Softmax function has been used where the sum of all output probabilities for all classes is 1.

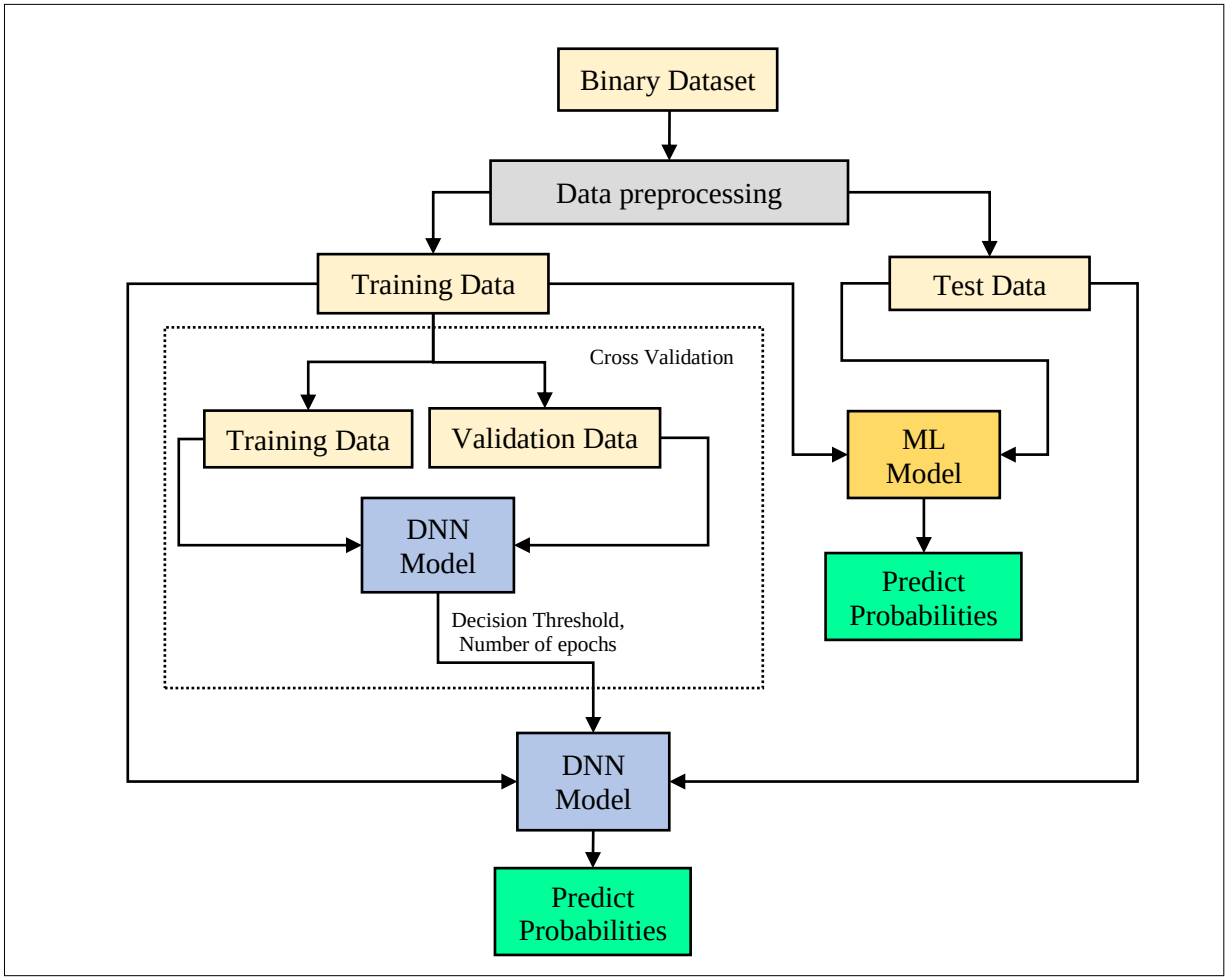

Figure 6.1: Procedure Workflow for comparison of proposed deep learning-based system with machine learning based system for binary datasets

6.2 Multi-Class Classification

Multi-class financial dataset used in our research work is also an imbalanced dataset and the experiments have been performed on it without modifying it i.e., without performing any undersampling or oversampling on it as it has been observed from the results of experiments performed using binary datasets that the hidden patterns may get lost on modifying the data.

DNN model has been trained on the training data and early stopping has been used to monitor the validation data's performance to avoid overfitting of the model. Since, Softmax has been used in output layer due to multi class data, which makes sure that sum of all output probabilities is equal to one. Hence, thresholding concept is not applicable in this case. Hence, no cross validation has been used for optimizing the decision threshold and number of epochs as done in case of binary classification.

Loss functions utilized with Deep Neural Network (DNN) for multi-classification are CEL and FL (without α parameter). As other loss functions like W-CEL, W-FL have been specifically designed for binary classification problems i.e., positive, and negative examples to increase the weightage of one class by downweighing the other class in the batch of samples, hence have not been utilized with this multi-class dataset. Also, RFL and WH-RFL use threshold values to down weight the easily classified examples, which is not applicable in multi-class problems where the sum of all output probabilities is 1.

The proposed deep learning-based fraud detection system has been compared to existing machine learning models which include the K Nearest Neighbor (KNN), the Decision Tree (DT), Light Gradient Boosting Machine (LGBM) and Random Forest (RF) except for Logistic Regression (LR) which is applicable to only binary class problems.

All machine learning algorithms have also been trained on imbalanced training data. The whole procedure adopted for comparison of the proposed system with existing for multi-class dataset has been shown in Figure 6.2.

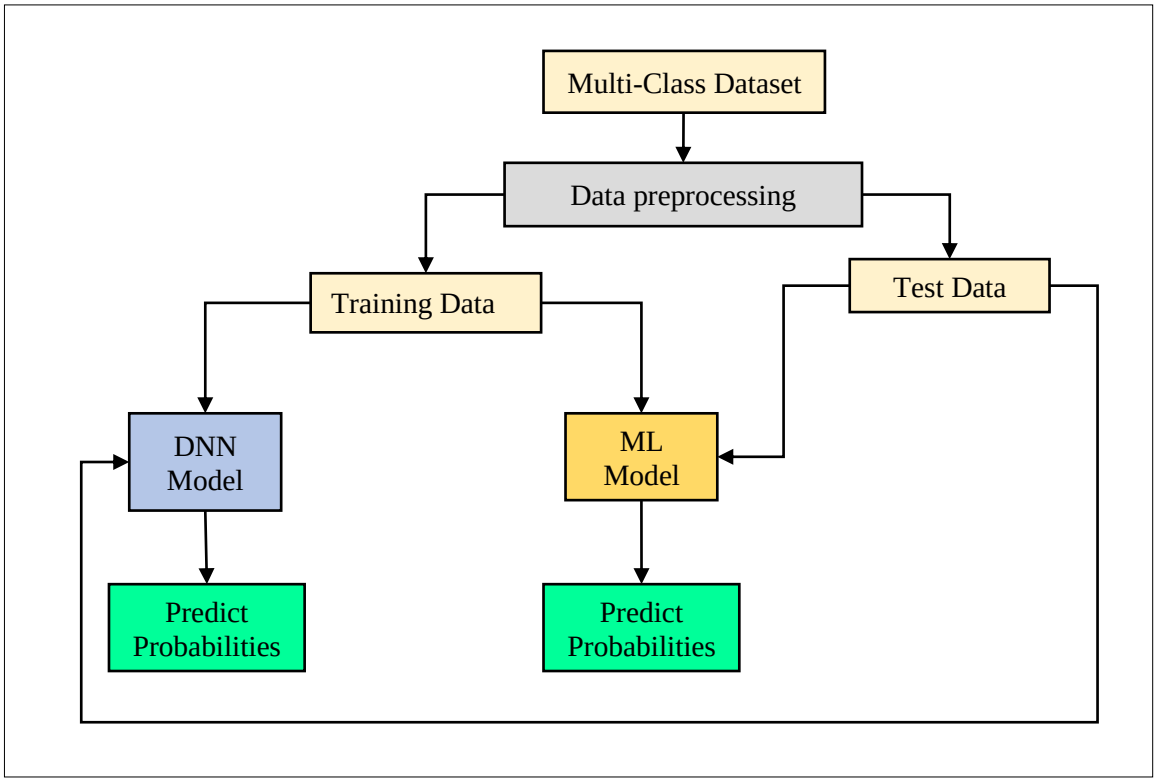

Figure 6.2: Procedure Workflow for comparison of proposed deep learning-based system with machine learning based system for multi-class dataset

Chapter-7

Results and Discussions

7.1 Results Analysis

The experiments were performed with Google Colab [111]. DNNs were implemented using Keras [112] and Tensorflow [113] and machine learning models were implemented using Scikit-learn [114] library. The experimental results have been summarized as per the following sections. The results produced by DNN by using various class imbalance methods have been discussed using binary datasets. Then, comparison of same with machine learning models has been done. In the last section, analysis of results produced by the DNN model using the third multi class financial dataset has also been done to validate our proposed model on multi class problem.

7.1.1 Selection of Thresholding Criteria for Binary Classification

Even though machine learning algorithms can predict a probability or scoring of class membership, this can only be achieved by mapping the values to a predefined threshold value which is usually 0.5.

However, when data is imbalanced, this threshold value gives poor performance. Thus, Closest to (0,1); Youden Index J; and max- G-Mean are three thresholding criteria which have taken into consideration while optimizing the threshold of DNN in case of imbalanced data. The most suitable one was selected among them which was Closest to (0,1).

For demonstration purposes, the DNN model has been trained using the CEL function. The results obtained by applying all thresholding criteria and default threshold after first epoch of training are compiled in Table 7.1. For both datasets, the decision threshold

calculated using the Closest-to-(0,1) method gave equal weightage to both the TPR and TNR and hence was chosen for our model. The results have already been published in [115-117].

Table 7.1

Results produced by thresholding criteria

Dataset	**Thresholding Criteria**	**Threshold**	**TPR**	**TNR**	**G-Mean**	**AUROC**
IEEE CIS	Default Threshold	0.5	0.3252	**0.9970**	0.5694	0.8915
	Youden Index J	0.0499	0.7768	0.8598	0.8172	
	Max G-Mean	0.0468	0.7864	0.8495	**0.8174**	
	Closest to (0,1)	0.0449	**0.7931**	0.8423	0.8173	
European Credit Card	Default Threshold	0.5	0.2237	**0.1000**	0.4730	0.8108
	Youden Index J	0.1049	0.6974	0.9686	0.8219	
	Max G-Mean	0.0798	0.7500	0.9043	**0.8235**	
	Closest to (0,1)	0.0704	**0.7763**	0.8531	0.8138	

Also, ROC curves generated using probabilities calculated for validation data after the first epoch using all three thresholding criteria and default threshold have been visualized in Figures 7.1 and 7.2.

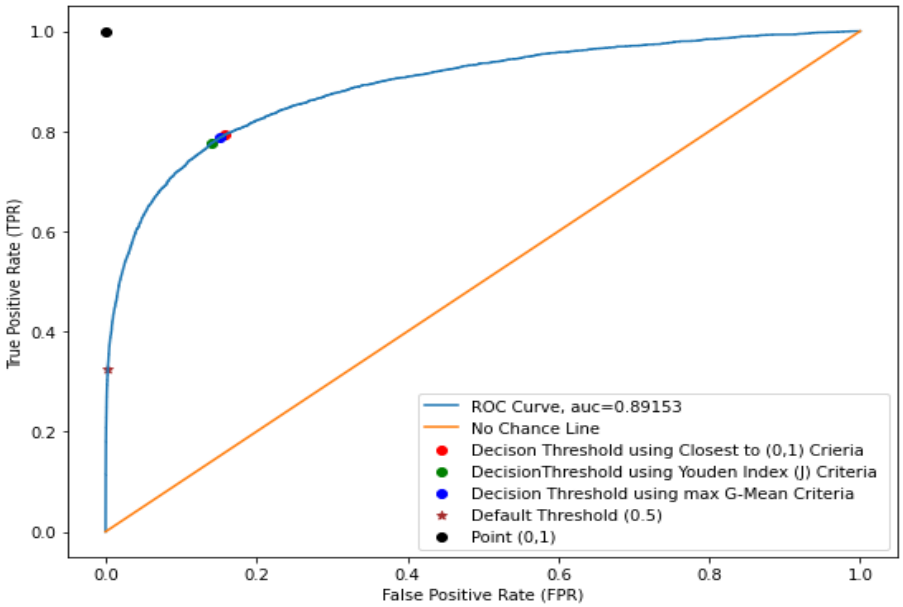

Figure 7.1: ROC curve for IEEE CIS Dataset Results after first epoch

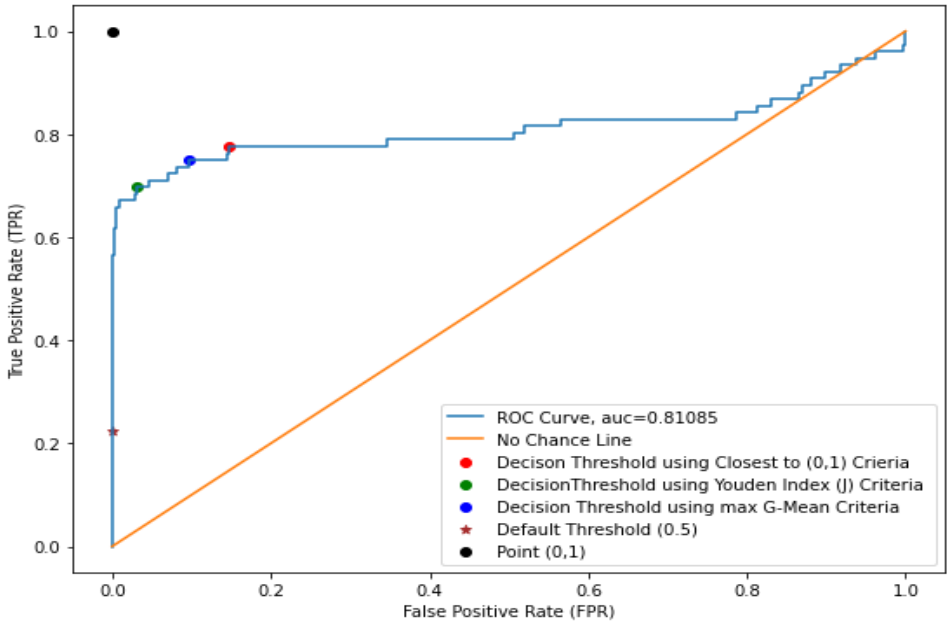

Figure 7.2: ROC curve for European Credit Card Dataset Results after first epoch

7.1.2 DNN Results using Class Imbalance Methods for Binary Datasets

The DNN results of all class imbalance methods have been presented in Table 7.2 for IEEE CIS dataset and in Table 7.3 for European Credit Card Dataset.

TABLE 7.2

IEEE CIS Dataset Results

Method		**Threshold**	**Epochs**	**TPR**	**TNR**	**G-Mean**	**AUROC**	**Accuracy**
Data Level	**RUS**	0.5	95	0.8914	0.8613	0.8762	0.9448	0.8623
	ROS	0.5	93	0.8270	0.9809	0.9007	0.9627	0.9755
Algorithm Level	**CEL**	0.0141	60	0.9114	0.8935	0.9024	0.9651	0.8941
	FL	0.1596	44	0.9148	0.8829	0.8987	0.9649	0.8840
	RFL	0.1537	42	0.9122	0.8881	0.9000	0.9648	0.8889
	α **-FL** (α = 0.10)	0.0869	39	0.9102	0.8781	0.8940	0.9625	0.8792
	α **-FL** (α = 0.25)	0.1296	30	0.9064	0.8741	0.8901	0.9605	0.8753
	α **-FL** (α = 0.50)	0.1489	53	0.9163	0.8847	0.9004	0.9665	0.8858
	α **-FL** (α = 0.75)	0.2189	36	0.9124	0.8871	0.8997	0.9641	0.8880
	α **-FL** (α = 0.90)	0.3021	30	0.9129	0.8648	0.8885	0.9602	0.8665
	W-CEL	0.2922	38	0.9129	0.8725	0.8925	0.9596	0.8740
	W-FL	0.3958	41	0.9160	0.8635	0.8894	0.9611	0.8653
	WH-RFL	0.1790	53	**0.9168**	0.8882	0.9024	0.9669	0.8892
Hybrid	**RUS-ROS**	0.3002	53	0.8609	0.9350	0.8972	0.9604	0.9201

TABLE 7.3

European Credit Card Dataset Results

Method		Threshold	Epochs	TPR	TNR	G-Mean	AUROC	Accuracy
Data Level	**RUS**	0.5	586	0.8632	0.9809	0.9201	0.9666	0.9807
	ROS	0.5	95	0.8632	0.9910	0.9249	0.9623	0.9908
Algorithm Level	**CEL**	0.0003	27	0.9053	0.9215	0.9133	0.9741	0.9215
	FL	0.0613	21	0.9158	0.9041	0.9099	0.9670	0.9041
	RFL	0.0595	67	0.9053	0.9441	0.9245	0.9789	0.9441
	α **-FL** (α = 0.10)	0.0192	205	0.9053	0.9486	0.9267	0.9689	0.9485
	α **-FL** (α = 0.25)	0.0496	36	0.8947	0.9570	0.9253	0.9733	0.9569
	α **-FL** (α = 0.50)	0.0613	21	0.9158	0.9041	0.9099	0.9670	0.9041
	α **-FL** (α = 0.75)	0.0867	74	**0.9263**	0.9347	0.9305	0.9779	0.9347
	α **-FL** (α = 0.90)	0.1212	49	**0.9263**	0.9322	0.9292	0.9746	0.9322
	W-CEL	0.0835	24	0.9158	0.9067	0.9112	0.9784	0.9067
	W-FL	0.2412	50	**0.9263**	0.8877	0.9068	0.9740	0.8877
	WH-RFL	0.0726	52	**0.9263**	0.9312	0.9287	0.9776	0.9312
Hybrid	**RUS-ROS**	0.1023	17	0.9053	0.9210	0.9131	0.9774	0.9210

Results of the experiments revealed that the algorithm level methods performed better when dealing with the imbalanced data. They also maintained the high value of TPR even in the datasets with high imbalance.

Results Analysis for Algorithm Level Methods using DNN

The results were produced using all loss functions under study. The test data's results were generated using the optimal threshold calculation. The results for the focal loss function were also checked for the various values of the balancing factor (α) and without balancing factor as well. Modulating factor (γ= 2) is same for all loss functions utilizing it.

From the experimental results, an inverse relation has been found between the optimal decision threshold and class imbalance present in the data. The same has been visualized in Figure 7.3. Also, the relationship between α and decision threshold. Higher the value of α for minority or positive class, higher is the value of decision threshold as observed in case of FL. Thus, the threshold got adjusted itself by modifying the learning of the model.

For both datasets, TPR achieved by WH-RFL is maximum but at the cost of slight increase in false positives. W-FL and α-FL have obtained equal TPR as that of WF-RFL in case of European Credit Card dataset. Thus, the proposed loss function was able to classify more fraud transactions correctly but at the cost of misclassify the genuine transactions. This research is the first study to show how to maximize the TPR by optimizing the decision threshold.

`

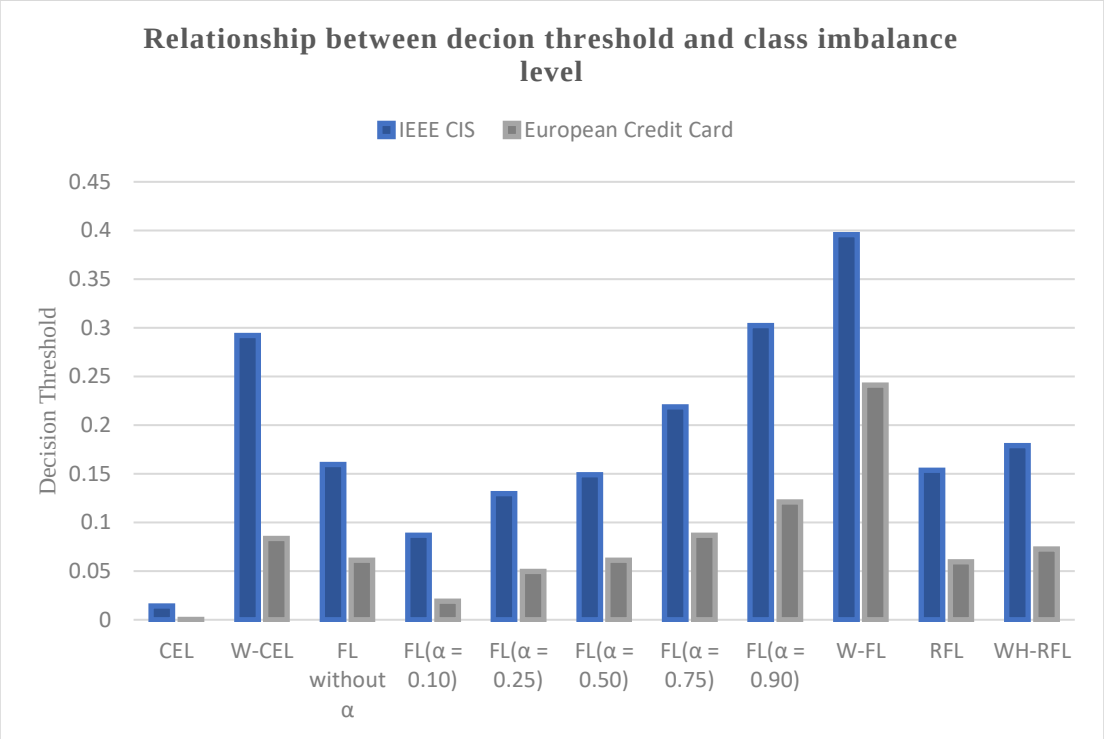

Figure 7.3: Relationship between Decision Threshold and Class Imbalance Level

7.1.3 Comparison of results of Proposed System with Existing Systems for Binary Datasets

The implementation of all loss functions used in algorithm level methods has also been done for LGBM model and the results of DNN model have been compared with the LGBM model and have been summarized in Table 7.4 for IEEE CIS dataset and in Table 7.5 for European Credit Card dataset.

Table 7.4

DNN vs LGBM for IEEE CIS Dataset

Loss Function	Model	TPR	TNR (Fixed)	G-Mean	AUROC	Accuracy
CEL	DNN	0.9114	0.8935	0.9024	0.9651	0.8941
	LGBM	0.9030		0.8982	0.9625	0.8938
FL	DNN	0.9148	0.8829	0.8987	0.9649	0.8840
	LGBM	0.9097		0.8962	0.9616	0.8838
RFL	DNN	0.9122	0.8881	0.9648	0.9648	0.8889
	LGBM	0.8932		0.8970	0.9589	0.8884
α -FL (α = 0.10)	DNN	0.9102	0.8781	0.8940	0.9625	0.8792
	LGBM	0.9131		0.8954	0.9615	0.8838
α -FL (α = 0.25)	DNN	0.9064	0.8741	0.8901	0.9605	0.8753
	LGBM	**0.9148**		0.8954	0.9628	0.8755
α -FL (α = 0.50)	DNN	0.9163	0.8847	0.9004	0.9666	0.8858
	LGBM	0.9061		0.8954	0.9607	0.8855
α -FL (α = 0.75)	DNN	0.9124	0.8871	0.8997	0.9641	0.8880
	LGBM	0.8984		0.8927	0.9588	0.8875
α -FL (α = 0.90)	DNN	0.9129	0.8648	0.8885	0.9602	0.8665
	LGBM	0.8897		0.8772	0.9545	0.8657
W-CEL	DNN	0.9129	0.8725	0.8925	0.9596	0.8740
	LGBM	0.8911		0.8818	0.9511	0.8732
W-FL	DNN	0.9160	0.8635	0.8894	0.9611	0.8653
	LGBM	0.8945		0.8789	0.9496	0.8646
WH-RFL	**DNN**	**0.9168**	0.8882	0.9024	0.9669	0.8892
	LGBM	0.8819		0.8851	0.9541	0.8880

Table 7.5

DNN vs LGBM for European Credit Card Dataset

Loss Function	Model	TPR	TNR (Fixed)	G-Mean	AUROC	Accuracy
CEL	DNN	0.9053	0.9215	0.9133	0.9741	0.9215
	LGBM	0.8842		0.9027	0.9488	0.9214
FL	DNN	0.9158	0.9041	0.9099	0.9670	0.9041
	LGBM	**0.8947**		0.8994	0.9376	0.9040
RFL	DNN	0.9053	0.9441	0.9245	0.9789	0.9441
	LGBM	0.8842		0.9137	0.9537	0.9440
FL ($\alpha = 0.10$)	DNN	0.9053	0.9486	0.9267	0.9688	0.9485
	LGBM	0.8842		0.9158	0.9410	0.9485
FL ($\alpha = 0.25$)	DNN	0.8947	0.9570	0.9253	0.9733	0.9569
	LGBM	0.8737		0.9144	0.9357	0.9568
FL ($\alpha = 0.50$)	DNN	0.9158	0.9041	0.9099	0.9670	0.9041
	LGBM	**0.8947**		0.8994	0.9287	0.9040
FL ($\alpha = 0.75$)	**DNN**	**0.9263**	0.9347	0.9305	0.9779	0.9347
	LGBM	0.8842		0.9091	0.9345	0.9346
FL ($\alpha = 0.90$)	**DNN**	**0.9263**	0.9322	0.9292	0.9746	0.9322
	LGBM	0.8737		0.9025	0.9273	0.9321
W-CEL	DNN	0.9158	0.9067	0.9112	0.9784	0.9067
	LGBM	0.8269		0.8659	0.8750	0.9066
W-FL	**DNN**	**0.9263**	0.8877	0.9068	0.9740	0.8877
	LGBM	0.8218		0.8541	0.8717	0.8876
WH-RFL	**DNN**	**0.9263**	0.9312	0.9287	0.9776	0.9312
	LGBM	0.8842		0.9311	0.9444	0.9074

It has been observed that DNN model has outperformed LGBM by achieving high TPR value at the same TNR value. Also, the LGBM with class weighted loss functions like W-CEL, W-FL and WH-RFL have not achieved good TPR values as compared to DNN. Thus, deep learning is more efficient as compared to gradient boosting in terms of handling the class imbalance by modifying its learning where we can assign more weightage to minority class to get high TPR.

The results generated for DNN using CEL have been selected as a baseline for comparison with all machine models since we are not altering the loss function or learning of the model to give more weightage to fraud transactions.

The probabilities for test datasets have been obtained by all learning models. The ROC curves for the machine learning models are derived from the test data probabilities. The ROCs curves for both datasets are presented in the Figures 7.4 & 7.5. The performance metrics used in this research work are decision threshold dependent except AUROC. Hence, we have chosen the FPR (1-TNR) value achieved by the DNN model as a cutoff point for a particular dataset and then we calculated the TPR values achieved by all machine learning models on that dataset from their ROC curves. Then, further metrics like G-Mean and accuracy were calculated. Results for both datasets are presented in Tables 7.6 and 7.7.

Table 7.6

Comparison of results of all models for IEEE CIS dataset

Model	**TPR**	**TNR (Fixed)**	**G-Mean**	**AUROC**	**Accuracy**
DT	0.6633	0.8935	0.7698	0.8069	0.8854
KNN	0.8130		0.7264	0.8897	0.8907
LR	0.7014		0.7917	0.8804	0.8868
RF	0.8855		0.8895	0.9476	0.8932
LGBM	0.9030		0.8982	0.9623	0.8938
DNN	**0.9114**		**0.9024**	**0.9651**	**0.8941**

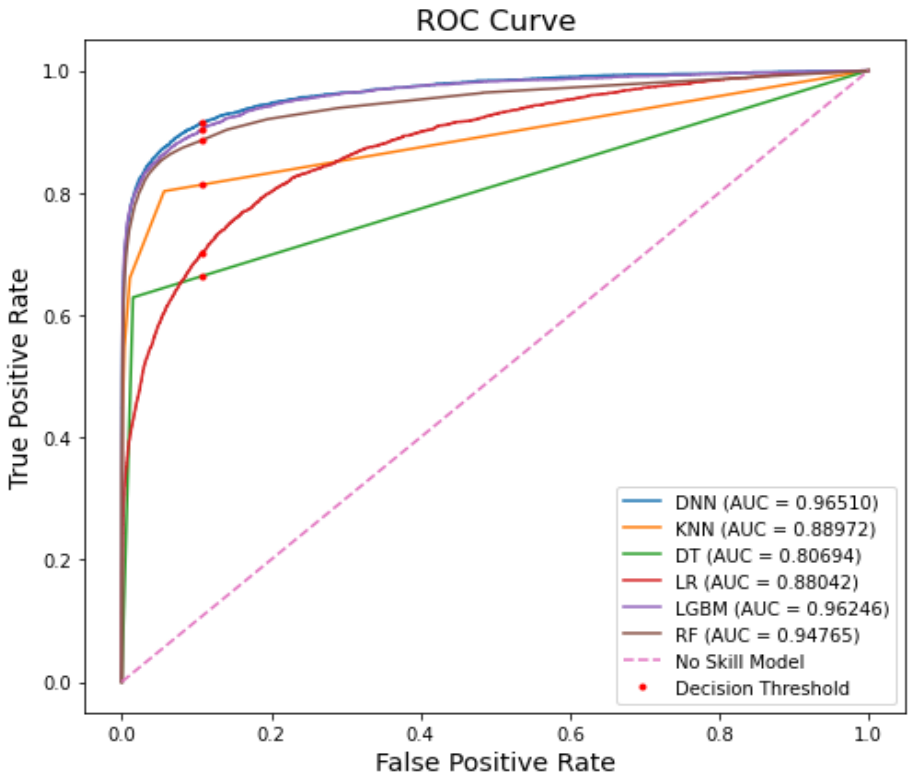

Figure 7.4: ROC curves for IEEE CIS dataset

Table 7.7

Comparison of results of all models for European Credit Card dataset

Model	**TPR**	**TNR (Fixed)**	**G-Mean**	**AUROC**	**Accuracy**
DT	0.7089	0.9215	0.8082	0.8419	0.9211
KNN	0.8155		0.8699	0.8999	0.9213
LR	0.8737		0.8973	0.9564	0.9214
RF	0.8611		0.8908	0.9244	0.9214
LGBM	0.8842		0.9027	0.9488	0.9214
DNN	**0.9053**		**0.9133**	**0.9741**	**0.9215**

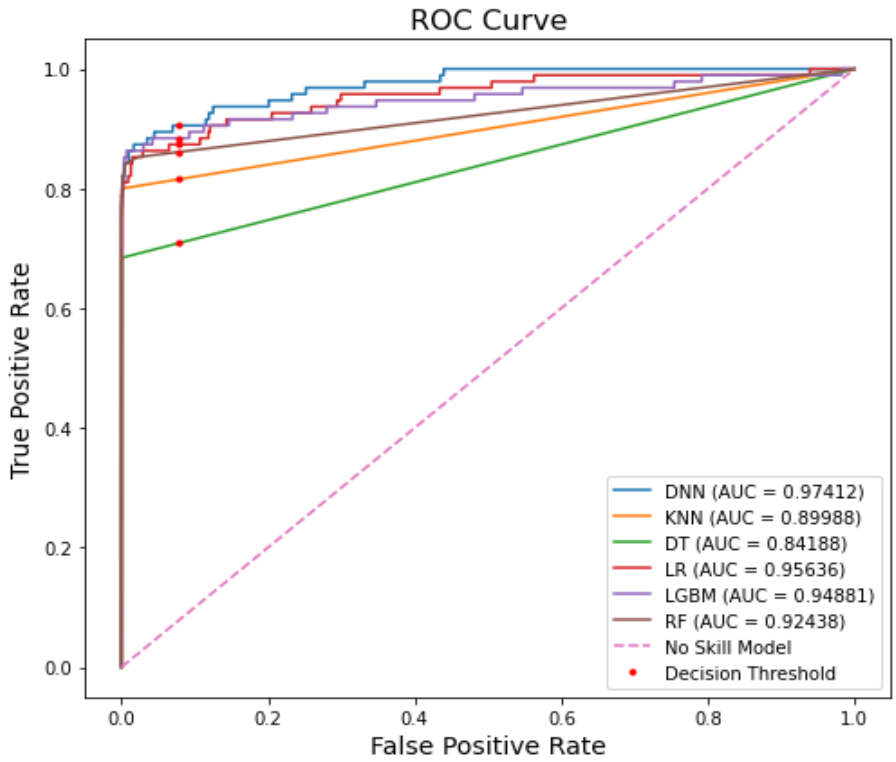

Figure 7.5: ROC curves for European Credit Card dataset

From experiment results, it is evident that the DNN model (using CEL) has achieved maximum AUROC, TPR, G-Mean and accuracy scores among all learning models by keeping FPR and TNR fixed for all both datasets. Hence, the DNN model can detect frauds more efficiently even in imbalanced datasets.

7.1.4 Multi-Class Financial Dataset Results

The results for the multi-class financial dataset using CEL and FL has been shown in Table 7.8. FL has achieved same performance as that of CEL with a smaller number of epochs of model training. The results have been calculated using machine learning algorithms as well (Table 7.9). The results for LGBM have been calculated using by default loss function i.e., multi-class log loss/cross entropy. Thus, DNN, KNN and RF have achieved equivalent performance.

TABLE 7.8

Multi-Class Financial Dataset Results using DNN

Model	**Loss Function**	**Epochs**	**Recall (Regular Transaction)**	**Recall (Global Fraud)**	**Recall (Local Fraud)**	**G-Mean**	**Accuracy**
DNN	CEL	37	1	1	1	1	1
DNN	FL	31	1	1	1	1	1

TABLE 7.9

Multi-Class Financial Dataset Results using machine learning algorithms

Model	**Recall (Regular Transaction)**	**Recall (Global Fraud)**	**Recall (Local Fraud)**	**G-Mean**	**Accuracy**
KNN	1	1	1	1	1
RF	1	1	1	1	1
DT	0.9999	1	0.8333	0.9410	0.9999
LGBM	0.9999	1	1	0.9999	0.9999

Chapter-8

Conclusion and Future Work

A system to detect frauds in online transactions that uses deep learning has been proposed in this study. The class imbalance problem has been resolved by optimizing the threshold in binary datasets. The results of the study indicate that the proposed method can outperform data level methods as hidden patterns can get lost when data is being modified. Thus, high TPR can be achieved without modifying the dataset and the overall performance of the system can also be maintained without much compromising the TNR rate.

This research work has utilized the Reduced Focal Loss function (RFL), a novel loss function that was modified to achieve high TPR. The relationship between the decision threshold and the class imbalance level has also been studied. The proposed fraud detection system based on deep learning is also more efficient than other machine learning techniques like KNN, LR, LGBM, DT, and RF in terms of detecting fraud.

With the help of large datasets, the proposed method can improve the performance of fraud detection systems without altering the data. The proposed system is also applicable to multi class fraud detection problem. The proposed method can be used in areas such as anomaly detection, disease detection, and healthcare. It can also be explored using other deep learning models.

In the future, the proposed system will be utilized with other class balancing methods which can detect fraud transactions by not decreasing the system's performance in terms of correctly classifying genuine transactions. The hybrid methods by combining the algorithm-level and data-level methods will be utilized to increase the efficiency of the proposed system.

Various loss functions specifically designed for binary classification problem like W-CEL, W-FL, RFL, WH-RFL, etc. will also be utilized with the proposed system for multi-class datasets by splitting them into multiple binary datasets by either using one-vs-rest or one-vs-one approach so that the importance of one class can be enhanced by downweighing the others in the dataset.

References

[1] S. Mason and N. Bohm, "Banking and fraud," *Comput. Law Secur. Rev.*, vol. 33, no. 2, pp. 237–241, Apr. 2017.

[2] V. Sapovadia, "Financial Inclusion, Digital Currency, and Mobile Technology," in *Handbook of Blockchain, Digital Finance, and Inclusion, Volume 2*, Elsevier, 2018, pp. 361–385.

[3] M. Carminati, R. Caron, F. Maggi, I. Epifani, and S. Zanero, "BankSealer: A decision support system for online banking fraud analysis and investigation," *Comput. Secur.*, vol. 53, pp. 175–186, 2015.

[4] A. Eshghi and M. Kargari, "Introducing a new method for the fusion of fraud evidence in banking transactions with regards to uncertainty," *Expert Syst. Appl.*, vol. 121, pp. 382–392, May 2019.

[5] Bolton and Hand, "Statistical Fraud Detection: A Review," *Stat. Sci.*, vol. 17, no. 3, pp. 235–249, 2002.

[6] S. Kovach and W. V. Ruggiero, "Online banking fraud detection based on local and global behavior," *in Proc. of the Fifth International Conference on Digital Society, Guadeloupe, France*, 2011, pp. 166–171.

[7] RBI Annual Report 2021, Available: https://rbi.org.in/scripts/AnnualReportPublications.aspx?Id=1319

[8] S. Liu, J. McGree, Z. Ge, and Y. Xie, "Classification methods," in *Computational and Statistical Methods for Analysing Big Data with Applications*, Elsevier, 2016, pp. 7–28.

[9] Chang, J. J., "An analysis of advance fee fraud on the internet", *Journal of Financial Crime*, Vol.15, no.1, 2008, pp.71-81.

[10] Sahu, K. R., & Dubey, J., "A survey on phishing attacks", *International Journal of Computer Applications*, Vol. 88, no.10, 2014, pp.42-45.

[11] Yar, M., & Steinmetz, K. F., "Cybercrime and society". Sage, 2019.

[12] Noufidali, V. M., Thomas, J. S., & Jose, F. A., "E-auction frauds-a survey", *International Journal of Computer Applications*, Vol. 61, no.14, 2013, pp.41-45.

[13] Gragido, W., & Pirc, J., "Cybercrime and espionage: An analysis of subversive multi-vector threats", Newnes, 2011.

[14] Brenner, S. W., “Cybercrime and the law: Challenges, issues, and outcomes”, UPNE, 2012.

[15] Subramanya, S. R., & Yi, B. K.,“Digital signatures”, *IEEE Potentials*, Vol. 25, no.2, pp. 5-8, 2006.

[16] Biegelman, M. T., & Bartow, J. T., “Executive roadmap to fraud prevention and internal control: Creating a culture of compliance”, *John Wiley & Sons*, 2012.

[17] V. Khattri and D. K. Singh, “Parameters of automated fraud detection techniques during online transactions,” *J. Financ. Crime*, vol. 25, no. 3, pp. 702–720, 2018.

[18] Abdallah, A., Maarof, M. A., & Zainal, A., “Fraud detection system: A survey”, *Journal of Network and Computer Applications*, Vol. 68, 2016, pp.90-113.

[19] Moore, R., “Cybercrime: Investigating high-technology computer crime”, Routledge, 2014.

[20] West, J., & Bhattacharya, M., “Intelligent financial fraud detection: a comprehensive review”, *Computers & security*, Vol. 57, 2016, pp.47-66.

[21] Zhang, X., Han, Y., Xu, W., & Wang, Q., “HOBA: A novel feature engineering methodology for credit card fraud detection with a deep learning architecture”, *Information Sciences*, 2019, pp.1-15.

[22] Fiore, U., De Santis, A., Perla, F., Zanetti, P., & Palmieri, F., “Using generative adversarial networks for improving classification effectiveness in credit card fraud detection”, *Information Sciences*, Vol. 479, 2019, pp. 448-455.

[23] Chen, J. I. Z., & Lai, K. L., “Deep Convolution Neural Network Model for Credit-Card Fraud Detection and Alert”, *Journal of Artificial Intelligence*, Vol. 3, no. 2, 2021, pp. 101-112.

[24] Sanober, S., Alam, I., Pande, S., Arslan, F., Rane, K. P., Singh, B. K., ... & Shabaz, M., “An enhanced secure deep learning algorithm for fraud detection in wireless communication”, Wireless Communications and Mobile Computing, 2021.

[25] Suvarna, R., & Kowshalya, A. M., "Credit Card Fraud Detection Using Deep Learning Techniques", *Journal of Web Engineering & Technology*, Vol. 7, no. 1, 2020, pp. 30-47.

[26] Miikkulainen, R., Liang, J., Meyerson, E., Rawal, A., Fink, D., Francon, O., ... & Hodjat, B., "Evolving deep neural networks", *In Artificial intelligence in the age of neural networks and brain computing*, Academic Press, 2019, pp. 293-312.

[27] Dornadula, V. N., & Geetha, S., "Credit card fraud detection using machine learning algorithms", *Procedia computer science*, Vol. 165, 2019, pp. 631-641.

[28] Khodabakhshi, M., & Fartash, M., "Fraud detection in banking using knn (k-nearest neighbor) algorithm", *In International Conf. on Research in Science and Technology*, 2016, pp.26-34.

[29] Brijain, M., Patel, R., Kushik, M. R., & Rana, K., "A survey on decision tree algorithm for classification", *International Journal of Engineering Development and Research*, Vol. 2, 2014, pp.1-5.

[30] Sahin, Y., & Duman, E., "Detecting credit card fraud by ANN and logistic regression", *International Symposium on Innovations in Intelligent Systems and Applications*, IEEE, 2011, pp. 315-319.

[31] G. Biau and E. Scornet, "A random forest guided tour," *Test (Madr.)*, vol. 25, no. 2, pp. 197–227, 2016.

[32] G. Ke *et al.*, "LightGBM: A highly efficient gradient Boosting Decision Tree," *Advances in neural information processing systems*, vol. *30*, pp.3146-3154, 2017.

[33] B. Li and D. Pi, "Learning deep neural networks for node classification," *Expert Syst. Appl.*, vol. 137, pp. 324–334, Dec. 2019.

[34] G. Hale and J. Lopez, "Monitoring banking system connectedness with big data," *J. Econom.*, Apr. 2019.

[35] A. Ng, "CS229 course notes: Deep learning", Stanford University, Fall 2018.

[36] M. Arya and H. Sastry G, "DEAL – 'Deep Ensemble ALgorithm' Framework for Credit Card Fraud Detection in Real-Time Data Stream with Google

TensorFlow," *Taylor & Francis*, 05-Jul-2020. Available: https://www.tandfonline.com/doi/abs/10.1080/23080477.2020.1783491.

[37] R. Setiono and H. Liu, "Feature extraction via Neural networks," *SpringerLink*, 01-Jan-1998. [Online]. Available: https://link.springer.com/chapter/10.1007/978-1-4615-5725-8_12.

[38] R. Setiono and H. Liu, "Feature extraction via Neural networks," *SpringerLink*, 01-Jan-1998. [Online]. Available: https://link.springer.com/chapter/10.1007/978-1-4615-5725-8_12.

[39] S. Pouyanfar *et al.*, "A survey on deep learning: Algorithms, techniques, and applications," *ACM Comput. Surv.*, vol. 51, no. 5, pp. 1–36, 2019.

[40] S. Dara and P. Tumma, "Feature extraction by using deep learning: A survey," in *2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA)*, 2018.

[41] S. Dargan, M. Kumar, M. Ayyagari and G. Kumar, "A Survey of Deep Learning and Its Applications: A New Paradigm to Machine Learning", *Archives of Computational Methods in Engineering*, vol. 27, no. 4, pp. 1071-1092, 2019. Available: 10.1007/s11831-019-09344-w.

[42] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[43] D. A. Bashar, "Survey on evolving deep learning neural network architectures," *December 2019*, vol. 2019, no. 2, pp. 73–82, 2019.

[44] F. Shaheen, B. Verma and M. Asafuddoula, "Impact of Automatic Feature Extraction in Deep Learning Architecture," 2016 International Conference on Digital Image Computing: Techniques and Applications (DICTA), Gold Coast, QLD, 2016, pp. 1-8, doi: 10.1109/DICTA.2016.7797053.

[45] J. Johnson and T. Khoshgoftaar, "Survey on deep learning with class imbalance", *Journal of Big Data*, vol. 6, no. 1, 2019.

[46] J. M. Johnson and T. M. Khoshgoftaar, "Deep Learning and Thresholding with Class-Imbalanced Big Data," *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, Boca Raton, FL, USA, 2019, pp. 755-762, doi: 10.1109/ICMLA.2019.00134.

[47] Kanika and J. Singla, "A Survey of Deep Learning based Online Transactions Fraud Detection Systems," *2020 International Conference on Intelligent Engineering and Management (ICIEM)*, London, United Kingdom, 2020, pp. 130-136, doi: 10.1109/ICIEM48762.2020.9160200.

[48] M. Kukar and I. Kononenko, "Cost-Sensitive Learning with Neural Networks", *ECAI*, 1998.

[49] T. Lin, P. Goyal, R. Girshick, K. He and P. Dollár, "Focal Loss for Dense Object Detection", *arXiv.org*, 2020. [Online]. Available: https://arxiv.org/abs/1708.02002.

[50] X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri, and R. M. Summers, "ChestX-Ray8: Hospital-Scale Chest X-Ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[51] R. Qin, K. Qiao, L. Wang, L. Zeng, J. Chen and B. Yan, "Weighted Focal Loss: An Effective Loss Function to Overcome Unbalance Problem of Chest X-ray14", *IOP Conference Series: Materials Science and Engineering*, vol. 428, p. 012022, 2018. Available: 10.1088/1757-899x/428/1/012022.

[52] N. Sergievskiy and A. Ponamarev, "Reduced Focal Loss: 1st Place Solution to xView object detection in Satellite Imagery", *arXiv.org*, 2020.

[53] J. Johnson and T. Khoshgoftaar, "Medicare fraud detection using neural networks", *Journal of Big Data*, vol. 6, no. 1, 2019.

[54] T. Peterson, M. Papeş, and J. Soberón, "Rethinking receiver operating characteristic analysis applications in ecological niche modeling," *Ecol. Modell.*, vol. 213, no. 1, pp. 63–72, 2008.

[55] B. Song, G. Zhang, W. Zhu and Z. Liang, "ROC operating point selection for classification of imbalanced data with application to computer-aided polyp detection in CT colonography", *International Journal of Computer Assisted Radiology and Surgery*, vol. 9, no. 1, pp. 79-89, 2013. Available: 10.1007/s11548-013-0913-8.

[56] I. Unal, "Defining an Optimal Cut-Point Value in ROC Analysis: An Alternative Approach", *Computational and Mathematical Methods in Medicine*, vol. 2017, pp. 1-14, 2017. Available: 10.1155/2017/3762651.

[57] W. J. Youden, “Index for rating diagnostic tests,” Cancer, vol. 3, no. 1, pp. 32–35, 1950.

[58] G. Kwon, J. Ryu, J. Oh, J. Lim, B.-kyeong Kang, C. Ahn, J. Bae, and D. K. Lee, “Deep learning algorithms for detecting and visualising intussusception on plain abdominal radiography in children: a retrospective multicenter study,” *Nature News*, 16-Oct-2020. [Online]. Available: https://www.nature.com/articles/s41598-020-74653-1. [Accessed: 12-Nov-2020].

[59] J. T. Hancock and T. M. Khoshgoftaar, “Survey on categorical data for neural networks,” *J. Big Data*, vol. 7, no. 1, 2020.

[60] G. Chandrashekar and F. Sahin, “A survey on feature selection methods,” *Comput. Electr. Eng.*, vol. 40, no. 1, pp. 16–28, 2014.

[61] X. Wan, “Influence of feature scaling on convergence of gradient iterative algorithm,” *J. Phys. Conf. Ser.*, vol. 1213, p. 032021, 2019.

[62] C. Feng, W. Hongyue, N. Lu, T. Chen, H. He, Y. Lu and X. M Tu, “Log-transformation and its implications for data analysis,” *Shanghai Arch. Psychiatry*, vol. 26, no. 2, pp. 105–109, 2014.

[63] O. N. Keene, “The log transformation is special,” *Stat. Med.*, vol. 14, no. 8, pp. 811–819, 1995.

[64] I. B. Mohamad and D. Usman, “Standardization and its effects on K-means clustering algorithm,” *Res. J. Appl. Sci. Eng. Technol.*, vol. 6, no. 17, pp. 3299–3303, 2013.

[65] U. Fiore, A. D. Santis, F. Perla, P. Zanetti, and F. Palmieri, “Using generative adversarial networks for improving classification effectiveness in credit card fraud detection,” *Information Sciences*, vol. 479, pp. 448–455, 2017.

[66] Z. Zhang, X. Zhou, X. Zhang, L. Wang and P. Wang, "A Model Based on Convolutional Neural Network for Online Transaction Fraud Detection", *Security and Communication Networks*, vol. 2018, pp. 1-9, 2018. Available: 10.1155/2018/5680264.

[67] J. A. Gómez, J. Arévalo, R. Paredes, and J. Nin, “End-to-end neural network architecture for fraud scoring in card payments,” *Pattern Recognition Letters*, vol. 105, pp. 175–181, 2018.

[68] J. Jurgovsky, M. Granitzer, K. Ziegler, S. Calabretto, P.-E. Portier, L. He-Guelton, and O. Caelen, “Sequence classification for credit-card fraud detection,” *Expert Systems with Applications*, vol. 100, pp. 234–245, 2018.

[69] A. Chouiekh and E. H. I. E. Haj, “ConvNets for Fraud Detection analysis,” *Procedia Computer Science*, vol. 127, pp. 133–138, 2018.

[70] A. Pumsirirat and L. Yan, “Credit Card Fraud Detection using Deep Learning based on Auto-Encoder and Restricted Boltzmann Machine,” *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 1, 2018.

[71] N. Abroyan, “Neural Networks for Financial Market Risk Classification,” *Frontiers in Signal Processing*, vol. 1, no. 2, 2017.

[72] A. Wiese and C. Omlin, “Credit Card Transactions, Fraud Detection, and Machine Learning: Modelling Time with LSTM Recurrent Neural Networks,” *Innovations in Neural Information Paradigms and Applications Studies in Computational Intelligence*, pp. 231–268, 2009.

[73] K. Fu, D. Cheng, Y. Tu, and L. Zhang, “Credit Card Fraud Detection Using Convolutional Neural Networks,” *Neural Information Processing Lecture Notes in Computer Science*, pp. 483–490, 2016.

[74] S. Wang, C. Liu, X. Gao, H. Qu, and W. Xu, “Session-Based Fraud Detection in Online E-Commerce Transactions Using Recurrent Neural Networks,” *Machine Learning and Knowledge Discovery in Databases Lecture Notes in Computer Science*, pp. 241–252, 2017.

[75] A. Shen, R. Tong, and Y. Deng, “Application of Classification Models on Credit Card Fraud Detection,” *2007 International Conference on Service Systems and Service Management*, 2007.

[76] E. L. Paula, M. Ladeira, R. N. Carvalho, and T. Marzagao, “Deep Learning Anomaly Detection as Support Fraud Investigation in Brazilian Exports and Anti-Money Laundering,” *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2016.

[77] Z. Kazemi and H. Zarrabi, “Using deep networks for fraud detection in the credit card transactions,” *2017 IEEE 4th International Conference on Knowledge-Based Engineering and Innovation (KBEI)*, 2017.

[78] N. Abroyan, “Convolutional and recurrent neural networks for real-time data classification,” *2017 Seventh International Conference on Innovative Computing Technology (INTECH)*, 2017.

[79] Y. Heryadi and H. L. H. S. Warnars, “Learning temporal representation of transaction amount for fraudulent transaction recognition using CNN, Stacked LSTM, and CNN-LSTM,” *2017 IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom)*, 2017.

[80] X. Lp, W. Yu, T. Luwang, J. Zheng, X. Qiu, J. Zhao, L. Xia, and Y. Li, “Transaction Fraud Detection Using GRU-centered Sandwich-structured Model,” *2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design ((CSCWD))*, 2018.

[81] C. Wang, Y. Wang, Z. Ye, L. Yan, W. Cai, and S. Pan, “Credit Card Fraud Detection Based on Whale Algorithm Optimized BP Neural Network,” *2018 13th International Conference on Computer Science & Education (ICCSE)*, 2018.

[82] P. Zheng, S. Yuan, X. Wu, J. Li, and A. Lu, “One-Class Adversarial Nets for Fraud Detection,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 1286–1293, 2019.

[83] Y. Ando, H. Gomi and H. Tanaka, "Detecting Fraudulent Behavior Using Recurrent Neural Networks", *Computer Security Symposium*, 11-13 October 2016, *Pdfs.semanticscholar.org*, 2016. [Online]. Available: https://www.semanticscholar.org/paper/Detecting-Fraudulent-Behavior-Using-Recurrent-Ando-Gomi/5d318371549831a05825de6e4eae4461538aeb9b.

[84] Y. Lu, "Deep neural networks and fraud detection", *DIVA*, 2017. [Online]. Available: http://uu.diva-portal.org/smash/record.jsf?pid=diva2%3A1150344&dswid=-3078.

[85] M. Renström and T. Holmsten, "Fraud Detection on Unlabeled Data with Unsupervised Machine Learning", *http://kth.diva-portal.org/*, 2018.

[86] M. Schreyer, T. Sattarov, D. Borth, A. Dengel and B. Reimer, "Detection of Anomalies in Large Scale Accounting Data using Deep Autoencoder Networks", *arXiv.org*, 2018. [Online]. Available: https://arxiv.org/abs/1709.05254.

[87] H. Choi, E. Jang and A. A. Alemi, "WAIC, but Why? Generative Ensembles for Robust Anomaly Detection", *Arxiv.org*, 2019. [Online]. Available: https://arxiv.org/abs/1810.01392.

[88] F. Ghobadi and M. Rohani, "Cost sensitive modeling of credit card fraud using neural network strategy," *2016 2nd International Conference of Signal Processing and Intelligent Systems (ICSPIS)*, Tehran, 2016, pp. 1-5.

[89] T. T. Nguyen, H. Tahir, M. Abdelrazek, and A. Babar, "Deep learning methods for credit card fraud detection," *arXiv [cs.LG]*, 2020.

[90] R. Y. Gupta, S. S. Mudigonda, and P. K. Baruah, "A comparative study of using various machine learning and deep learning-based fraud detection models for universal health coverage schemes," *Int. j. eng. trends technol.*, vol. 69, no. 3, pp. 96–102, 2021.

[91] C. Elkan, "Evaluating Classifiers", January 20, 2012. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.233.2746&rep=rep1&type=pdf

[92] J. Fan, S. Upadhye, and A. Worster, "Understanding receiver operating characteristic (ROC) curves: Canadian Journal of Emergency Medicine," Cambridge Core, 21-May-2015.

[93] Kanika, J. Singla and Nikita, "Comparing ROC Curve based Thresholding Methods in Online Transactions Fraud Detection System using Deep Learning," 2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), 2021, pp. 9-12, doi: 10.1109/ICCCIS51004.2021.9397167.

[94] J. L. Leevy, T. M. Khoshgoftaar, R. A. Bauder, and N. Seliya, "A survey on addressing high-class imbalance in big data," *J. Big Data*, vol. 5, no. 1, 2018.

[95] IEEE Computational Intelligence Society, "IEEE-CIS fraud detection data," 2019**.** [Online]. Available: https://www.kaggle.com/c/ieee-fraud-detection/data

[96] Machine Learning Group-ULB, “Credit card fraud detection dataset,” 2018. [Online]. Available: https://www.kaggle.com/mlg-ulb/creditcardfraud

[97] M. M. Suarez-Alvarez, D.-T. Pham, M. Y. Prostov, and Y. I. Prostov, “Statistical approach to normalization of feature vectors and clustering of mixed datasets,” *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 468, no. 2145, pp. 2630–2651, 2012.

[98] R. H. H. Groenwold, I. R. White, A. R. T. Donders, J. R. Carpenter, D. G. Altman, and K. G. M. Moons, “Missing covariate data in clinical research: when and when not to use the missing-indicator method for analysis,” *Canadian Medical Association Journal*, vol. 184, no. 11, pp. 1265–1269, 2012.

[99] M. Herland, T. M. Khoshgoftaar, and R. A. Bauder, “Big Data fraud detection using multiple medicare data sources,” *Journal of Big Data*, vol. 5, no. 1, 2018.

[100] M. Tharmakulasingam, C. Topal, A. Fernando and R. L. Ragione, "Backward Feature Elimination for Accurate Pathogen Recognition Using Portable Electronic Nose," *2020 IEEE International Conference on Consumer Electronics (ICCE)*, 2020, pp. 1-5, doi: 10.1109/ICCE46568.2020.9043043.

[101] Robust Scaler, Online Available: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.RobustScaler.html

[102] R. Taylor, “Interpretation of the correlation coefficient: A basic review,” *J. Diagn. Med. Sonogr.*, vol. 6, no. 1, pp. 35–39, 1990.

[103] M. Schreyer, T. Sattarov, C. Schulze, B. Reimer, and D. Borth, “Detection of accounting anomalies in the latent space using adversarial autoencoder neural networks,” *arXiv [cs.LG]*, 2019.

[104] Multi-class Financial Fraud detection dataset. [Online]. Available: https://github.com/GitiHubi/deepAD

[105] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,’’ *Journal of Machine Learning Research*, 2014, 1929-1958.

[106] M. D. Richard and R. P. Lippmann, “Neural Network Classifiers Estimate Bayesian a posteriori Probabilities,” *Neural Computation,* vol. 3, no. 4, pp. 461–483, 1991.

[107] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” *arXiv.org,* 02-Mar-2015. [Online]. Available: https://arxiv.org/abs/1502.03167.

[108] K. He, X. Zhang, S. Ren, and J. Sun, “Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification,” *2015 IEEE International Conference on Computer Vision (ICCV),* 2015.

[109] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *International Conference for Learning Representations, 3rd International Conference, ICLR,* 2015, arXiv:1412.6980

[110] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall, “Activation functions: Comparison of trends in practice and research for deep learning,” *arXiv [cs.LG]*, 2018.

[111] Google Colab. https://research.google.com/colaboratory/faq.html

[112] F. Chollet, Keras, 2015. https://keras.io

[113] M. Abadi et al., "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems", *arXiv.org,* 2020. [Online]. Available: https://arxiv.org/abs/1603.04467.

[114] Scikit-learn. https://scikit-learn.org/

[115] Kanika, J. Singla, A. Kashif Bashir, Y. Nam, N. UI Hasan, and U. Tariq, “Handling class imbalance in online transaction fraud detection,” *Computers Materials Continua (CMC),* vol. 70, no. 2, pp. 2861–2877, 2022.

[116] Kanika and J. Singla, “A novel framework for online transaction fraud detection system based on deep neural network,” *Journal of Intelligent & Fuzzy Systems (JIFS),* pp. 1–11, 2022.

[117] Kanika and J. Singla, "Class Balancing Methods for Fraud Detection using Deep Learning," *2022 Second International Conference on Artificial Intelligence and Smart Energy (ICAIS),* 2022, pp. 395-400, doi: 10.1109/ICAIS53314.2022.9742836.

Appendix 1: Source Code

1. Implementation of Deep Neural Network (DNN) for IEEE CIS dataset

```
# create model architecture
model = Sequential()
model.add(Dense(512, kernel_initializer=he_uniform(seed=42), bias_initializer=i
nitializers.Zeros(), input_shape=(558,)))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.3, seed= 42))

model.add(Dense(256, kernel_initializer=he_uniform(seed=42), bias_initializer=i
nitializers.Zeros()))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.3, seed= 42))

model.add(Dense(1, kernel_initializer=he_uniform(seed=42), use_bias= True, bia
s_initializer= initializers.Constant(b)))
model.add(Activation('sigmoid'))

model.compile(optimizer=Adam(learning_rate = 0.0001), loss= cross_entropy_los
s())
```

2. Implementation of DNN model for European Credit Card dataset

```
# create model architecture
model = Sequential()
model.add(Dense(512, kernel_initializer=he_uniform(seed=42), bias_initializer=i
nitializers.Zeros(), input_shape=(30,)))
model.add(BatchNormalization())
model.add(Activation('relu'))
```

```
model.add(Dropout(0.3, seed= 42))

model.add(Dense(256, kernel_initializer=he_uniform(seed=42), bias_initializer=i
nitializers.Zeros()))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.3, seed= 42))

model.add(Dense(1, kernel_initializer=he_uniform(seed=42), use_bias= True, bia
s_initializer= initializers.Constant(b)))
model.add(Activation('sigmoid'))

model.compile(optimizer=Adam(learning_rate = 0.0001), loss= cross_entropy_los
s())
```

3. Implementation of Deep Neural Network (DNN) for multi-Class Financial dataset

```
# create model architecture
model = Sequential()
model.add(Dense(256, kernel_initializer=he_uniform(seed=42), bias_initializer=i
nitializers.Zeros(), input_shape=(618,)))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.3, seed= 42))

model.add(Dense(128, kernel_initializer=he_uniform(seed=42), bias_initializer=i
nitializers.Zeros()))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.3, seed= 42))
```

```
model.add(Dense(3, kernel_initializer=he_uniform(seed=42), use_bias= True, bia
s_initializer= initializers.Constant(b)))
model.add(Activation('softmax'))

model.compile(optimizer=Adam(learning_rate = 0.0001), loss= cross_entropy_los
s())
```

4. Implementation of various Machine learning models for IEEE CIS dataset

#K Nearest Neighbour (KNN)

```
knn = KNeighborsClassifier(n_neighbors = 5, metric = 'euclidean')
```

#Logistic Regression (LR)

```
logreg=LogisticRegression(penalty = 'l2' )
```

#LGBM (Light Gradient Boosting Machine)

```
lgbm_params = {'learning_rate':0.05, 'boosting_type':'gbdt',
        'metric': 'cross_entropy_loss',
        'num_leaves':100,
        'max_depth':10,
        'seed': 42,
        'early_stopping': 20
         }
```

#RF (Random Forest)

```
rfc = RandomForestClassifier(n_estimators=100, criterion ='gini',  max_depth =
        'None', min_samples_split ='2', min_samples_leaf='1')
```

#DT (Decision Tree)

```
dt_classifier = DecisionTreeClassifier(criterion ='gini', splitter= 'best', max_depth =
'None', min_samples_split ='2', min_samples_leaf='1')
```

5. Implementation of various Machine learning models for European Credit Card dataset

```
# K Nearest Neighbour (KNN)
knn = KNeighborsClassifier(n_neighbors = 5, metric = 'euclidean')

# Logistic Regression (LR)
logreg=LogisticRegression(penalty = 'l2' )

# LGBM (Light Gradient Boosting Machine)
lgbm_params = {'learning_rate':0.05, 'boosting_type':'gbdt',
        'metric': 'cross_entropy_loss',
        'num_leaves':100,
        'max_depth':10,
        'seed' : 42,
        'early_stopping' : 20
         }

# RF (Random Forest)
rfc = RandomForestClassifier(n_estimators=100, criterion ='gini',  max_depth =
  'None', min_samples_split ='2', min_samples_leaf='1')

# DT (Decision Tree)
dt_classifier = DecisionTreeClassifier(criterion ='gini', splitter= 'best', max_depth =
  'None', min_samples_split ='2', min_samples_leaf='1')
```

6. Implementation of various Machine learning models for European Credit Card dataset

```
# K Nearest Neighbour (KNN)
knn = KNeighborsClassifier(n_neighbors = 5, metric = 'euclidean')

# LGBM (Light Gradient Boosting Machine)
lgbm_params = {'learning_rate':0.05, 'boosting_type':'gbdt',
          'num_leaves':100,
          'max_depth':10,
          'seed' : 42,
          'objective': 'multiclass',
          'num_class':3,
          'early_stopping' : 20,
           'seed' : 42
```

```
            }
    # RF (Random Forest)
rfc = RandomForestClassifier(n_estimators=100, criterion ='gini',  max_depth =
    'None', min_samples_split ='2', min_samples_leaf='1')

    # DT (Decision Tree)
 dt_classifier = DecisionTreeClassifier(criterion ='gini', splitter= 'best', max_depth =
    'None', min_samples_split ='2', min_samples_leaf='1')
```

7. Implementation of various loss functions to be used with DNN model

```
    # Binary Cross Entropy Loss function
     def binary_cross_entropy():

        def binary_cross_entropy_fixed(y_true, y_pred):
            pt_1 = tf.where(tf.equal(y_true, 1), y_pred, tf.ones_like(y_pred))
            pt_0 = tf.where(tf.equal(y_true, 0), y_pred, tf.zeros_like(y_pred))
            return -K.sum(K.log(K.epsilon()+pt_1))-K.sum(K.log(1. -
      pt_0 + K.epsilon()))
        return binary_cross_entropy_fixed

    # Categorical Cross Entropy Loss function

     def categorical_cross_entropy_loss():
        def cross_entropy_loss(y_true, y_pred):
            epsilon = K.epsilon()
            y_pred = K.clip(y_pred, epsilon, 1.0-epsilon)
            # Calculate cross entropy
            cross_entropy = -y_true*K.log(y_pred)
            # Sum the losses in mini_batch
            loss = K.sum(cross_entropy, axis=1)
            return loss
        return cross_entropy_loss
```

```
# Binary Focal Loss function
def focal_loss(gamma=2., alpha=.25):
    def focal_loss_fixed(y_true, y_pred):
        pt_1 = tf.where(tf.equal(y_true, 1), y_pred, tf.ones_like(y_pred))
        pt_0 = tf.where(tf.equal(y_true, 0), y_pred, tf.zeros_like(y_pred))
        return -K.sum(alpha * K.pow(1. - pt_1, gamma) * K.log(K.epsilon()+pt_1))-
K.sum((1-alpha) * K.pow( pt_0, gamma) * K.log(1. - pt_0 + K.epsilon()))
    return focal_loss_fixed

# Categorical Focal Loss function
def categorical_focal_loss(gamma=2.0):
    def focal_loss(y_true, y_pred):
        epsilon = K.epsilon()
        y_pred = K.clip(y_pred, epsilon, 1.0-epsilon)
        # Calculate cross entropy loss
        cross_entropy_loss = -y_true*K.log(y_pred)
        # Calculate focal loss
        loss = y_true * K.pow((1-y_pred), gamma) * cross_entropy_loss
        # Sum the losses in mini_batch
        loss = K.sum(loss, axis=1)
        return loss

    return focal_loss

# Weighted Cross Entropy Loss function
def weighted_binary_cross_entropy():
    def weighted_binary_cross_entropy_fixed(y_true, y_pred):
        p = K.sum(y_true)
        p = tf.cast(p, dtype='float32')
        equal_to_zero = tf.where(tf.equal(y_true, 0), tf.ones_like(y_true), tf.zeros_li
```

```
ke(y_true))
        n = K.sum(equal_to_zero)
        n = tf.cast(n, dtype='float32')
        pt_1 = tf.where(tf.equal(y_true, 1), y_pred, tf.ones_like(y_pred))
        pt_0 = tf.where(tf.equal(y_true, 0), y_pred, tf.zeros_like(y_pred))
        return - K.sum( ((p+n)/(p+K.epsilon())) * K.log(K.epsilon()+pt_1))-
K.sum( ((p+n)/(n+K.epsilon())) * K.log(1. - pt_0 + K.epsilon()))
    return weighted_binary_cross_entropy_fixed

# Weighted Focal Loss function
 def weighted_focal_loss(gamma=2.):
    def weighted_focal_loss_fixed(y_true, y_pred):
        p = K.sum(y_true)
        p = tf.cast(p, dtype='float32')
        equal_to_zero = tf.where(tf.equal(y_true, 0), tf.ones_like(y_true), tf.zeros_li
ke(y_true))
        n = K.sum(equal_to_zero)
        n = tf.cast(n, dtype='float32')
        pt_1 = tf.where(tf.equal(y_true, 1), y_pred, tf.ones_like(y_pred))
        pt_0 = tf.where(tf.equal(y_true, 0), y_pred, tf.zeros_like(y_pred))

        return -K.sum( ((p+n)/(p+K.epsilon())) * K.pow(1. -
 pt_1, gamma) * K.log(K.epsilon()+pt_1))-
K.sum( ((p+n)/(n+K.epsilon())) * K.pow( pt_0, gamma) * K.log(1. -
 pt_0 + K.epsilon()))

    return weighted_focal_loss_fixed

# Reduced Focal Loss function

 def reduced_focal_loss(gamma=2.):
    def reduced_focal_loss_fixed(y_true, y_pred):
```

```
        pt_1 = tf.where(tf.equal(y_true, 1), y_pred, tf.ones_like(y_pred))
        pt_0 = tf.where(tf.equal(y_true, 0), y_pred, tf.zeros_like(y_pred))
        pt_1_reduced_threshold = tf.where(tf.math.less(pt_1,
0.5), K.log(K.epsilon()+pt_1), K.pow(1. -
 pt_1, gamma) * K.log(K.epsilon()+pt_1))
        pt_0_reduced_threshold = tf.where(tf.math.less((1-pt_0), 0.5), K.log(1. -
 pt_0 + K.epsilon()), K.pow( pt_0, gamma) * K.log(1. - pt_0 + K.epsilon()))
        return - K.sum(pt_1_reduced_threshold) - K.sum(pt_0_reduced_threshold)
    return reduced_focal_loss_fixed
```

 # Weighted Hard Reduced Focal Loss function

```
def weighted_hard_reduced_focal_loss(gamma=2., weight1= 2, weight2= 0.5):

    def weighted_hard_reduced_focal_loss_fixed(y_true, y_pred:

        pt_1 = tf.where(tf.equal(y_true, 1), y_pred, tf.ones_like(y_pred))

        pt_0 = tf.where(tf.equal(y_true, 0), y_pred, tf.zeros_like(y_pred))

        pt_1_reduced_threshold = tf.where(tf.math.less(pt_1, 0.5), weight1 * K.log(
K.epsilon()+pt_1), K.pow(1. - pt_1, gamma) * K.log(K.epsilon()+pt_1))

        pt_0_reduced_threshold = tf.where(tf.math.less((1-pt_0), 0.5), weight2 *
K.log(1. - pt_0 + K.epsilon()), K.pow( pt_0, gamma) * K.log(1. -
 pt_0 + K.epsilon()))
        return - K.sum(pt_1_reduced_threshold) - K.sum(pt_0_reduced_threshold
return easy_focal_loss_and_weight_to_hard_pos_bce_fixed
```

8. Implementation of various loss functions to be used with LGBM model

#Binary Cross Entropy function

```
def cross_entropy_loss(y_pred, dtrain):
    t = dtrain.label

    def bce(x,t):
        p = 1/(1+np.exp(-x))
        p = np.clip(p, 1e-15, 1 - 1e-15)
        return -( t * np.log(p) + (1- t) * np.log(1. - p))

    partial_bce = lambda x: bce(x, t)
    gradbce = derivative(partial_bce, y_pred, n=1, dx=1e-6)
    hessbce = derivative(partial_bce, y_pred, n=2, dx=1e-6)

    return gradbce, hessbce

def cross_entropy_loss_eval_error(y_pred, dtrain):
    y_true = dtrain.label
    preds = 1/(1+np.exp(-y_pred))
    preds = np.clip(preds, 1e-15, 1 - 1e-15)
    loss = -(y_true * np.log(preds) + (1 - y_true) * np.log(1. - preds))
    sum_loss = np.sum(loss)

    return 'cross_entropy_loss', sum_loss, False

#Focal Loss function
def focal_loss(y_pred, dtrain, gamma):
    g = gamma
    t = dtrain.label

    def fl(x,t):
        p = 1/(1+np.exp(-x))
        p = np.clip(p, 1e-15, 1 - 1e-15)
```

```
        return -( t * (1. - p)**g * np.log(p) + (1 - t) * (p)**g * np.log(1. - p))

    partial_fl = lambda x: fl(x, t)
    gradfl = derivative(partial_fl, y_pred, n=1, dx=1e-6)
    hessfl = derivative(partial_fl, y_pred, n=2, dx=1e-6)

    return gradfl , hessfl

def focal_loss_eval_error(y_pred, dtrain, gamma):

 g = gamma
 y_true = dtrain.label
 preds = 1/(1+np.exp(-y_pred))
 preds = np.clip(preds, 1e-15, 1 - 1e-15)

 loss = -(y_true * (1. - preds)**g * np.log(preds) + (1 -
y_true) * (preds)**g * np.log(1. - preds))

 sum_loss = np.sum(loss)
 return 'focal_loss', sum_loss, False

#Weighted Binary Cross Entropy function
    def weighted_cross_entropy_loss(y_pred, dtrain):

        t = dtrain.label
        def wbce(x,t):
            p = 1/(1+np.exp(-x))
            p = np.clip(p, 1e-15, 1 - 1e-15)
        cp = np.count_nonzero(t==1)
        cn = np.count_nonzero(t==0)
            return -((cp+cn)/cp * t * np.log(p) + (cp+cn)/cn * (1- t) * np.log(1. - p))
        partial_bce = lambda x: bce(x, t)
```

```
        gradbce = derivative(partial_bce, y_pred, n=1, dx=1e-6)
        hessbce = derivative(partial_bce, y_pred, n=2, dx=1e-6)
        return gradbce, hessbce

    def weighted_cross_entropy_loss_eval_error(y_pred, dtrain):
        y_true = dtrain.label
        preds = 1/(1+np.exp(-y_pred))
        preds = np.clip(preds, 1e-15, 1 - 1e-15)
        cp = np.count_nonzero(t==1)
        cn = np.count_nonzero(t==0)

        loss = -( (cp+cn)/cp * y_true * np.log(preds) + (cp+cn)/cn * (1 -
y_true) * np.log(1. - preds))
        sum_loss = np.sum(loss)
        return 'weighted_cross_entropy_loss', sum_loss, False

    #Weighted Focal Loss function
    def weighted_focal_loss(y_pred, dtrain, gamma):
        g = gamma
        t = dtrain.label
        def fl(x,t):
            p = 1/(1+np.exp(-x))
            p = np.clip(p, 1e-15, 1 - 1e-15)
            cp = np.count_nonzero(t==1)
            cn = np.count_nonzero(t==0)
            return -( (cp+cn)/cp * t * (1. - p)**g * np.log(p) + (cp+cn)/cn * (1 -
            t) * (p)**g * np.log(1. - p))

        partial_fl = lambda x: fl(x, t)
        gradfl = derivative(partial_fl, y_pred, n=1, dx=1e-6)
        hessfl = derivative(partial_fl, y_pred, n=2, dx=1e-6)
```

```
        return gradfl , hessfl

def focal_loss_eval_error(y_pred, dtrain, gamma):

  g = gamma
  y_true = dtrain.label
  preds = 1/(1+np.exp(-y_pred))
  preds = np.clip(preds, 1e-15, 1 - 1e-15)
  cp = np.count_nonzero(t==1)
  cn = np.count_nonzero(t==0)
loss = -((cp+cn)/cp * y_true * (1. - preds)**g * np.log(preds) + (cp+cn)/cn * (1 -
 y_true) * (preds)**g * np.log(1. - preds))

  sum_loss = np.sum(loss)
  return 'focal_loss', sum_loss, False

     #Reduced Focal Loss function
      def reduced_focal_loss(y_pred, dtrain, gamma):
        g = gamma
        t = dtrain.label
        def fl(x,t):
            p = 1/(1+np.exp(-x))
            p = np.clip(p, 1e-15, 1 - 1e-15)
            return -( t * (1. - p)**g * np.log(p) + (1 - t) * (p)**g * np.log(1. - p))

        partial_fl = lambda x: fl(x, t)
        gradfl = derivative(partial_fl, y_pred, n=1, dx=1e-6)
        hessfl = derivative(partial_fl, y_pred, n=2, dx=1e-6)

        def bce(x,t):
            p = 1/(1+np.exp(-x))
            p = np.clip(p, 1e-15, 1 - 1e-15)
```

```
            return -( t * np.log(p) + (1- t) * np.log(1. - p))

        partial_bce = lambda x: bce(x, t)
        gradbce = derivative(partial_bce, y_pred, n=1, dx=1e-6)
        hessbce = derivative(partial_bce, y_pred, n=2, dx=1e-6)

        p = 1/(1+np.exp(-y_pred))
        p = np.clip(p, 1e-15, 1 - 1e-15)
        grad = np.where(p < 0.5, gradbce , gradfl)
        hess = np.where(p < 0.5, hessbce,  hessfl)

        return grad, hess

def reduced_focal_loss_eval_error(y_pred, dtrain, gamma):

  g = gamma
  y_true = dtrain.label
  preds = 1/(1+np.exp(-y_pred))
  preds = np.clip(preds, 1e-15, 1 - 1e-15)

  loss = -( np.where(np.less(preds,0.5), y_true * np.log(preds) +  (1-
 y_true) * np.log(1. - preds), y_true * (1. - preds)**g * np.log(preds) + (1 -
 y_true) * (preds)**g * np.log(1. - preds)) )

  sum_loss = np.sum(loss)
  return 'reduced_focal_loss', sum_loss, False

#Weighted Hard Reduced Focal Loss function

      def weighted_hard_reduced_focal_loss(y_pred, dtrain, gamma):
        g = gamma
        t = dtrain.label
```

```
        def fl(x,t):
            p = 1/(1+np.exp(-x))
            p = np.clip(p, 1e-15, 1 - 1e-15)
            return -( t * (1. - p)**g * np.log(p) + (1 - t) * (p)**g * np.log(1. - p))

        partial_fl = lambda x: fl(x, t)
        gradfl = derivative(partial_fl, y_pred, n=1, dx=1e-6)
        hessfl = derivative(partial_fl, y_pred, n=2, dx=1e-6)


        def hard_bce(x,t):
            p = 1/(1+np.exp(-x))
            p = np.clip(p, 1e-15, 1 - 1e-15)
            return -( 2* t * np.log(p) + 0.5 *(1- t) * np.log(1. - p))

        partial_bce = lambda x: bce(x, t)
        gradbce = derivative(partial_bce, y_pred, n=1, dx=1e-6)
        hessbce = derivative(partial_bce, y_pred, n=2, dx=1e-6)

        p = 1/(1+np.exp(-y_pred))
        p = np.clip(p, 1e-15, 1 - 1e-15)
        grad = np.where(p < 0.5, gradbce , gradfl)
        hess = np.where(p < 0.5, hessbce,  hessfl)

        return grad, hess

def weighted_hard_reduced_focal_loss_eval_error(y_pred, dtrain, gamma):
  g = gamma
  y_true = dtrain.label
  preds = 1/(1+np.exp(-y_pred))
  preds = np.clip(preds, 1e-15, 1 - 1e-15)
```

```
 loss = -( np.where(np.less(preds,0.5), 2 * y_true * np.log(preds) +  0.5 * (1-
y_true) * np.log(1. - preds), y_true * (1. - preds)**g * np.log(preds) + (1 -
y_true) * (preds)**g * np.log(1. - preds)) )
 sum_loss = np.sum(loss)
 return 'weighted_hard_reduced_focal_loss', sum_loss, False
```

9. **Implementation of procedure to optimize the threshold to handle class imbalance scenario in binary classification**

```
class RocCallback(Callback):

    def __init__(self,training_data,validation_data,patience=0):
        self.x = training_data[0]
        self.y = training_data[1]
        self.x_val = validation_data[0]
        self.y_val = validation_data[1]
        self.patience = patience
        self.best_weights = None

    def on_train_begin(self, logs=None):

        self.wait = 0
        self.stopped_epoch = 0
        self.best_vtpr = 0
        self.best_vtnr = 0
        self.best_epoch = 0
        return

    def on_epoch_begin(self, epoch, logs=None):
        return

    def on_epoch_end(self, epoch, logs=None):
```

```
        y_pred_val = self.model.predict(self.x_val)

        # Calculate roc curve
        fpr, tpr, thresholds_roc = roc_curve(y[val], y_pred_val, pos_label= 1)

        D = np.sqrt(pow(1-tpr, 2) + pow(fpr,2))
        ix = argmin(D)
        print('Distance from (0,1), D=%.4f is'  % (D[ix]))
        print('Best Threshold=%.4f, TPR=%.4f, TNR=%4f of validation data is'  % (
thresholds_roc[ix], tpr[ix], 1-fpr[ix]))

        vtpr  = tpr[ix]
        vtnr = 1-fpr[ix]

        if np.greater_equal(vtpr, self.best_vtpr):

           if np.greater(vtpr, self.best_vtpr):
              self.best_vtpr = vtpr
              self.best_vtnr = vtnr
              self.wait = 0

          self.best_weights = self.model.get_weights()
              self.best_epoch = epoch
              logs["best_epochs"] = self.best_epoch + 1

           if np.equal(vtpr, self.best_vtpr) and np.greater(vtnr, self.best_vtnr):
              self.best_vtpr = vtpr
              self.best_vtnr = vtnr
              self.wait = 0
              self.best_weights = self.model.get_weights()
              self.best_epoch = epoch
```

```
            logs["best_epochs"] = self.best_epoch + 1

        elif np.equal(vtpr, self.best_vtpr):
            self.wait += 1
            if self.wait >= self.patience:
                self.stopped_epoch = epoch
                self.model.stop_training = True
                print("Restoring model weights from the end of the best epoch.")
                self.model.set_weights(self.best_weights)

    else:
        self.wait += 1
        if self.wait >= self.patience:
            self.stopped_epoch = epoch
            self.model.stop_training = True
            print("Restoring model weights from the end of the best epoch.")
            self.model.set_weights(self.best_weights)

def on_batch_begin(self, batch, logs=None):
    return

def on_batch_end(self, batch, logs=None):
    return

def on_train_end(self, logs=None):
    if self.stopped_epoch > 0:
        print("Epoch %05d: early stopping" % (self.stopped_epoch + 1))
        print("Best Number of epochs are", self.best_epoch + 1)
    return
```

List of Publications

The list of works published during this thesis are summarized as below.

Peer reviewed international journal articles

- Kanika and J. Singla, "Online banking fraud detection system: A review," *International Journal of Advanced Trends in Computer Science Engineering*, vol. 8, no. 3, pp. 959–962, 2019.
- Kanika, J. Singla, A. Kashif Bashir, Y. Nam, N. UI Hasan, and U. Tariq, "Handling class imbalance in online transaction fraud detection," *Computers Materials Continua (CMC)*, vol. 70, no. 2, pp. 2861–2877, 2022.
- Kanika and J. Singla, "A novel framework for online transaction fraud detection system based on deep neural network," *Journal of Intelligent & Fuzzy Systems (JIFS)*, pp. 1–11, 2022.

Peer reviewed international conference papers

- Kanika and J. Singla, "A survey of deep learning based online transactions fraud detection systems," in *2020 International Conference on Intelligent Engineering and Management (ICIEM)*.
- Kanika, J. Singla, and Nikita, "Comparing ROC curve based thresholding methods in online transactions fraud detection system using deep learning," in *2021 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*.
- Kanika and J. Singla, "Class Balancing Methods for Fraud Detection using Deep Learning," in *2022 Second International Conference on Artificial Intelligence and Smart Energy (ICAIS)*.