Enhancing the Robustness of Credit Card Fraud Detection Systems Against Adversarial Attacks Using Machine Learning

Rathnayake R.M.C

219392J

MSc in Computer Science

Department of Computer Science and Engineering
Faculty of Engineering

University of Moratuwa
Sri Lanka

March 2025

Enhancing the Robustness of Credit Card Fraud Detection Systems Against Adversarial Attacks Using Machine Learning

Rathnayake R.M.C

219392J

Thesis/Dissertation submitted in partial fulfillment of the requirements for the degree MSc in Computer Science

Department of Computer Science and Engineering
Faculty of Engineering

University of Moratuwa
Sri Lanka

March 2025

DECLARATION

I declare that this is my own work and this thesis/dissertation does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other University or Institute of higher learning and to the best of my knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text. I retain the right to use this content in whole or part in future works (such as articles or books).

Signature: Date: 05/06/2025

The above candidate has carried out research for the PhD/MPhil/Masters thesis/dissertation under my supervision. I confirm that the declaration made above by the student is true and correct.

Name of Supervisor: Prof. Sanath Jayasena

Signature of the Supervisor: Date: 05th June 2025

DEDICATION

This research is dedicated to the countless individuals who have fallen victim to credit card fraud, whose experiences have fueled the urgency for robust and resilient fraud detection systems. It is also dedicated to the tireless efforts of researchers, practitioners, and financial institutions working diligently to protect consumers from these malicious activities. May this research contribute to the collective knowledge and efforts aimed at safeguarding the integrity of digital financial transactions.

ACKNOWLEDGEMENT

The successful completion of this research would not have been possible without the invaluable contributions and support of numerous individuals and organizations, I extend my deepest appreciation to my advisor, Prof. Sanath Jayasena, for providing unwavering guidance, mentorship, and encouragement throughout this research journey. Your expertise, patience, and belief in my abilities have been instrumental in shaping this work. Thank you for your invaluable contributions to this research.

I am profoundly humbled and grateful for the opportunity to have embarked on this journey, and I earnestly hope that the findings of this work will contribute to the advancement of knowledge and the betterment of society.

ABSTRACT

Credit card fraud detection systems are essential for protecting money transactions and stopping illegal use. The usefulness of these systems is seriously threatened by the increasing prevalence of adversarial assaults, as attackers try to alter input data in order to trick machine learning algorithms and evade detection. The problem of making credit card fraud detection systems more resilient to such hostile attacks is addressed in this study. It explores current defense tactics, analyses the status of adversarial attacks, and evaluates how they affect model performance. The study offers a comprehensive strategy that involves implementing several defense strategies, modelling hostile settings, and assessing their effectiveness using important performance indicators. The results show that the proposed approach greatly enhances the robustness of fraud detection models, effectively reducing the influence of adversarial manipulations

Credit card fraud detection systems face a significant challenge as adversarial attacks grow more complex, enabling attackers to alter data and avoid detection. By methodically evaluating defense tactics against these threats, this study seeks to improve fraud detection models. Various adversarial attack types, including hybrid, white-box, and black-box attacks, are simulated to identify weaknesses in current systems. Four defense mechanisms, namely, Neural Cleanse, Random Noise, General Adversarial Training, and Defensive Distillation are implemented and assessed. The results demonstrate that none of these methods offer total protection, although they improve model resilience to some degree. It backs up the claim that financial institutions do not yet have infallible defenses against hostile attacks. The study provides insightful information about enhancing fraud prevention by outlining the advantages and disadvantages of each strategy.

This study supports the development of more effective credit card fraud detection systems by offering actionable insights and a structured approach to designing and assessing protective strategies against adversarial threats. The findings have implications for financial institutions seeking to fortify their security posture and protect customers from fraudulent activities, fostering trust and confidence in digital financial transactions.

Keywords: Credit card fraud; Adversarial attacks; Robustness; Defense mechanisms; Machine learning; Security; Financial transactions; Fraud detection

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ACOFS	Ant Colony Optimization Feature Selection
AUC	Area Under the Curve
AUC-ROC	Area Under the ROC Curve
CNN	Convolutional Neural Network
DEFS	Differential Evolutionary Feature Selection
DNNs	Deep Neural Networks
ELM	Extreme Learning Method
FP	False Positive
FN	False Negative
GA	Genetic Algorithm
GAFS	Genetic Algorithm Feature Selection
GBM	Gradient Boosting Machine
GAT	Generative Adversarial Trainer
KNN	K-Nearest Neighbors
LR	Logistic Regression
LOF	Local Outlier Factor
LGBM	Light Gradient Boosting Machine
NB	Naive Bayes
PCA	Principal Component Analysis
PSOFS	Particle Swarm Optimization Feature Selection
RF	Random Forest
ROC	Receiver Operating Characteristic
RHSO	Rock Hyrax Swarm Optimization
SVM	Support Vector Machine
TN	True Negative
TP	True Positive

CHAPTER 1

INTRODUCTION

Credit card fraud continues to pose a serious threat to financial institutions and customers alike. To address this challenge, the financial sector is progressively adopting machine learning and various statistical analyzing methods as key tools in prevention of fraud. These include approaches such as supervised learning, unsupervised learning, and data mining. Supervised learning methods involve training models on past transaction data to help differentiate between genuine and fraudulent activity. Meanwhile, unsupervised learning focuses on detecting irregular spending patterns that diverge from usual behavior. Moreover, data mining is vital in detecting credit card fraud, as it helps reveal hidden patterns and relationships within massive sets of transaction data. This insight supports the detection of unusual activity and contributes to building effective fraud detection frameworks.

Integrating machine learning into fraud detection frameworks offers significant advantages, such as enhanced precision, fewer false positives, and the ability to identify fraudulent actions in real time. These benefits have led financial institutions to increasingly rely on such technologies to boost security and reduce financial losses. Nevertheless, even with their increasing accuracy, these systems remain vulnerable to adversarial attacks intentional alterations designed to mislead machine learning algorithms. In credit card transactions, for instance, attackers may slightly adjust transaction data to trick the system into incorrectly identifying fraudulent activity as legitimate, or the other way around. This not only results in financial harm but also undermines public confidence in the security of digital payment systems.

This research focuses on the escalating risk posed by adversarial attacks within credit card fraud detection systems. The primary aim is to enhance the resilience of these models, allowing them to reliably identify fraudulent transactions despite complex and constantly changing manipulation strategies.

1.1 Credit Card fraud

The execution of unlawful or dishonest transactions, such as unauthorized purchases, withdrawals, or fund transfers, without the card owner's agreement is referred to as credit card fraud. Through a variety of tactics, fraudsters exploit flaws in payment systems, frequently resulting in significant financial losses for both individuals and companies. These illicit actions can take many different forms, all of which are designed to avoid detection of the integrity of the transaction process. Some of the most common ways to perpetrate this kind of fraud are listed below.

1. Unauthorized Purchases

Credit card information can be stolen by fraudsters and used to make illegal purchases. Because there is no requirement for the cardholder to be physically present or produce

a signature for verification, this kind of fraud is especially common in online transactions.

2. Identity Theft

Identity theft happens when fraudsters obtain personal details such as Social Security numbers and use them to unlawfully open credit card accounts in another person's name. These accounts are then exploited to carry out unauthorized transactions and purchases.

3. Card-Not-Present (CNP) Transactions

Fraudsters steal credit card information and personal details to make unauthorized purchases online or over the phone. Because these transactions don't require the physical card for verification, preventing them can be especially challenging.

4. Counterfeit Cards

Criminals can produce fake credit cards by copying the details from genuine cards onto counterfeit ones. They then use these fake cards for in-person transactions, making them appear as though they are legitimate purchases.

5. Lost or Stolen Card Usage

When a credit card is lost or stolen, there is a window of opportunity for unauthorized individuals to do purchases. This vulnerable period, which is the time between the moment the card is taken and the time the rightful owner finds out and reports the loss, is commonly exploited by criminals seeking to carry out illicit actions.

6. Skimming

Credit card skimming is a method where special devices, known as skimmers, are used to steal information from the magnetic stripe of a credit card. These devices are discreetly placed on card readers at ATMs, fuel pumps, or retail checkout points. The captured data is then used to carry out unauthorized transactions.

7. Phishing and Social Engineering

Fraudsters employ fake emails, messages, or phone calls to manipulate individuals into disclosing their credit card details. Once acquired, this information is misused to carry out unauthorized transactions.

8. Adversarial attacks

Cybercriminals target weaknesses within credit card fraud detection systems by strategically crafting deceptive information designed to evade identification. These sophisticated attacks employ various approaches, including black-box, white-box, and hybrid strategies, to trick machine learning algorithms into incorrectly labeling fraudulent activities as genuine. This poses a significant threat to financial safety because standard fraud prevention methods may be ill-equipped to recognize these advanced forms of manipulation.

Credit card fraudulent transactions can lead to financial losses for both cardholders and financial institutions. Detecting and preventing such transactions require robust security measures, including advanced fraud detection systems, secure authentication methods, and prompt reporting and resolution processes.

1.1.1 Existing Approaches for the Credit Card Fraud Detection

A variety of cutting-edge technology and tactics are used in contemporary techniques to identify and halt fraudulent transactions with credit cards. These essential instruments are essential for preserving the integrity of financial transactions and shielding cardholders and financial institutions from unauthorized use. Important tactics now used to identify possibly fraudulent activities include:

1. Rule-Based Systems

Rule-based systems use predefined criteria and thresholds to identify potentially fraudulent transactions. These rules are typically grounded in past transaction patterns and behaviors that are considered normal for legitimate transactions. For instance, a rule might raise an alert if a transaction takes place in another location of cardholder's usual spending area.

2. Anomaly Detection

Anomaly detection focuses on recognizing irregularities or deviations in transaction data that differ from typical patterns. Machine learning algorithms analyze historical credit card transaction information to create a model of typical behavior. Any transaction that deviates considerably from this established norm may then be flagged for closer scrutiny.

3. Collaborative Filtering

Collaborative filtering techniques leverage collective user behavior data to identify potential fraud. By comparing an individual user's behavior with the behavior of the broader user community, anomalies can be detected.

4. Behavioral Analysis

The goal of behavioral analysis is to track the spending habits of specific consumers over time. Fraudulent activity may be indicated by departures from a user's customary behavior, such as abruptly big transactions or an unusually high frequency of transactions.

5. Monitor Real-Time Transactions

Many systems use real-time monitoring to evaluate transactions as they happen. This enables quick identification and response to suspicious activities, offering a proactive method for preventing fraud.

6. Geolocation and Device Analysis

Examining the geolocation of transactions and the devices involved can help identify irregularities. For instance, if a card is used in two different locations at the same time, it may trigger an alert.

7. Biometric Authentication

Certain systems use biometric authentication, like fingerprints or facial recognition, as an extra safeguard. Biometric data is difficult to fake, making the verification process more robust and secure.

8. Machine Learning Models

Supervised machine learning models are trained on labelled datasets that contain examples of both legitimate and fraudulent transactions. Decision trees, logistic regression, random forests, and other ensemble approaches are commonly used in these models. These algorithms get better at differentiating between real and fake activities as more data is provided.

Considering the current status of existing systems today, they have been able to curb fraud effectively and within limits. Evolving fraudulent activities come with a need for constant innovations and developments in other anti-fraud systems. These systems are equipped with advanced technologies designed to detect unauthorized transactions in credit cards. Effectual in securing financial transactions, such systems track every transaction in the particular credit card and try to detect any unusual pattern that indicates a possible detection. Most important for such a system are the machine-learning algorithms, which work increasingly on very large data sets to find signs and indications of fraud.

One reason for relying heavily on machine learning is the high frequency of credit card transactions. According to a Capital One research report from June 9, 2023, there are a staggering 104,274 transactions happening every minute [7].

Machine learning is crucial because it enables these systems to quickly process and analyze large volumes of transactions. As the financial landscape changes, these systems constantly enhance their ability to detect and prevent credit card fraud, making them essential tools in guarding against increasingly sophisticated fraudulent methods [12].

1.1.2 Adversarial Attacks

Adversarial attacks are deliberate methods used to mislead machine learning models by introducing small, often invisible changes to input data. These modifications are crafted to exploit weaknesses in how models make decisions, ultimately causing them to generate inaccurate predictions or classifications. At its core, an adversarial attack alters input data in subtle ways that exploit a model's sensitivity to small changes, modifications often imperceptible to humans, exposing critical weaknesses in the model's ability to remain reliable under such conditions.

Anant Jain [10] explores how sneaky tricks can confuse computer programs called neural networks. Imagine showing a computer a picture of a panda, and it confidently says it's a panda with 57.7% certainty. Now, add a little bit of tricky noise to the picture. Suddenly, the computer is convinced it's looking at a gibbon, and it's really sure about it, at 99.3%. It's like a puzzle for the computer, making it see things that aren't there. Surprisingly, to eyes, both pictures still look like pandas, and can't even notice the added noise. This shows that these tricks, which can't see, can make the computer completely misunderstand what it's looking at.

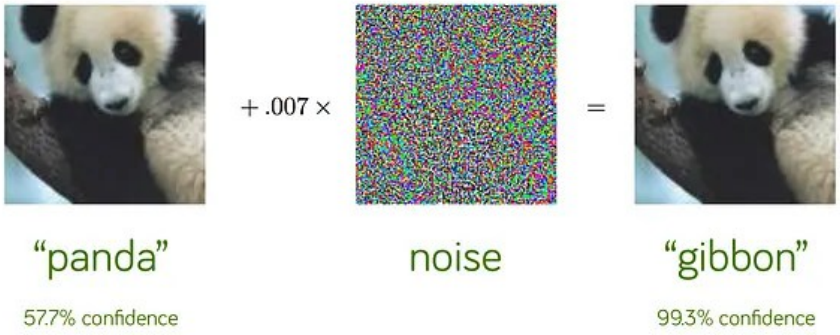

Figure 1.1

Example research for adversarial attack (source – [10]*)*

Note. In figure 1.1, the panda image is depicted after an adversarial attack, wherein it is identified as a gibbon with 99.3% confidence.

Here are some common types of adversarial attacks listed below

1. White-Box Attacks
2. Black-Box Attacks
3. Transfer Attacks
4. Evasion Attacks
5. Poisoning Attacks
6. Exploratory Attacks.
7. Physical Adversarial Attacks
8. Gradient-Based Attacks

Adversarial attacks remain a dynamic field of research, with ongoing efforts by experts to design stronger models and effective defense strategies aimed at reducing the impact of these deceptive manipulations [10].

1.2 Research Problem

The growing use of digital credit cards has made it more vital than ever to have strong systems that can find and stop fraud. These systems are key to keeping money safe and protecting users from unauthorized access. While regular computer learning methods have been good at finding fraud by looking at past information, they can be easily tricked by clever attacks. This is a big danger to how much they can trust these systems and how safe they are.

This research looks closely at the serious problem of these computer learning systems being attacked. These attacks intentionally change the information given to the system so it gets confused and misclassifies transactions. For example, when it comes to credit card fraud, online criminals might find weak spots in the system and make small changes to transaction details. This can cause the system to make mistakes and let fake transactions go through without being noticed.

The main challenge is to come up with strong ways to defend these fraud-finding systems so they can hold up against these attacks. Urgently requested better solutions that can both correctly spot real transactions and reduce the harm caused by these tricky attacks. Dealing with this problem is essential to make sure fraud detection systems are dependable and work well, especially as attackers come up with even smarter ways to try and fool them.

Key aspects to be explored in the research problem section include:

1. Understanding Adversarial Threats

Examine the features of adversarial attacks aimed at credit card fraud detection systems, emphasizing the weaknesses that attackers may take advantage of and the potential consequences of successful manipulations on the system.

2. Existing Defense Mechanisms

Assess the limitations of existing defense mechanisms used in fraud detection systems and evaluate their effectiveness in adversarial environments. Identify the gaps and areas where current methods can be improved.

1.3 Research aim and Objectives

1.3.1 Aim

The Aim of this study is to create and implement a solution that enhances the robustness of credit card fraud detection systems when faced with adversarial attacks.

1.3.2 Objectives

1. Design a Robust Credit Card Fraud Detection System.

This objective focuses on understanding and analyzing adversarial attacks that threaten credit card fraud detection. It involves studying various attack techniques, such as

adversarial examples, data poisoning, and evasion attacks, while also exploring existing defense mechanisms and their weaknesses. The insights gained will help in designing a more resilient fraud detection system.

2. Develop a Prototype for Fraud Detection with Adversarial Defense

The next step is to build a working prototype of a credit card fraud detection system that can withstand adversarial attacks. This will involve using machine learning techniques, adversarial training, and anomaly detection to strengthen the system. Key aspects of development include data preprocessing, feature engineering, model selection, and integrating defense strategies like adversarial training and model ensembling.

3. Evaluate the Effectiveness of the Developed System

The main goal is to check how well and how strongly new system works when it's faced with tricky attacks. To do this, we'll test it using both real transaction information and fake information that's been changed to try and fool it. I'll look at important measures like how often it's right, how precise it is, how well it finds all the fraud, and how hard it is for the attacks to trick it. I'll also compare it to other ways of finding fraud that are already out there to see if the system does a better job.

My research will show how hostile or tricky situations affect systems that try to spot credit card fraud, while also evaluating and strengthening defense mechanisms to ensure resilience against evolving threats.

CHAPTER 2

LITERATURE REVIEW

Fraud is when someone tricks others in a dishonest or illegal way to gain personal or financial advantages. It involves purposely breaking the law, rules, or policies to get money or benefits that are not allowed. There are already many reports, studies, and articles written about finding fraud and unusual activity in this area, and they are all available to the public.

Many studies have looked at how to find credit card fraud. For my paper, I want to see how well these existing fraud detection systems work when smart attackers are trying to trick them. This involves testing these models under conditions where they may face intentional attempts to deceive or manipulate their functionality, replicating real-world adversarial scenarios. The primary aim is to observe how these models perform and respond under such challenging conditions. Additionally, the research seeks to implement various defense mechanisms to improve the accuracy of fraud detection in the face of adversarial challenges. The focus is on developing and testing strategies that strengthen these models against fraudulent activities, ultimately enhancing their robustness and effectiveness.

As figure 2.1, the sequential steps include collecting credit card data, selecting relevant features, implementing a fraud detection model, simulating adversarial conditions for evaluation, implementing defense mechanisms, and subsequently evaluating the effectiveness of the deployed defense mechanisms.

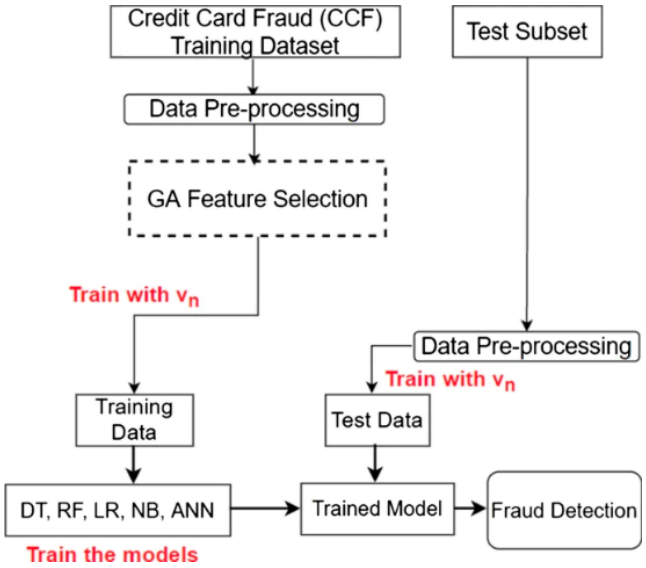

Figure 2.1

Existing approaches for Credit Card fraud identifying. (Source: [1])

2.1 Feature Selection for Unbalanced Data Set

A comprehensive study by Emmanuel Ileberi et al. [2] examined the use of feature selection and genetic algorithms (GA) in credit card fraud detection. The paper proposes a machine learning-based system that leverages a genetic algorithm for feature selection. For their research paper, they used different computer learning techniques to find fake credit card transactions. To make sure my system uses the best information, they applied a special method that helps it pick out the most useful details. And they tested how well their system works with real credit card fraud data from people in Europe.

This study looks at a dataset of 284,807 European credit card transactions that were gathered during two days in September 2013. Merely 0.172 percent of these transactions were found to be fraudulent. Along with additional information like the transaction time and amount, the dataset contains 30 features, referred to as V1 through V28.

Selecting the right features from the data is crucial when applying machine learning. Feature selection (FS) plays a vital role, as too many features in the dataset can hinder model performance. The selection of FS mechanism is based on the specific matter that they were trying to reach. Below are examples that demonstrate how using FS methods can increase the performance of selecting models.

The Genetic Algorithm (GA) is an advanced computational technique often used to solve problems more effectively. This algorithm has several key attributes that make it particularly useful for optimizing solutions.

Population: Genetic Algorithms (GAs) maintain a set of potential solutions known as a population.
Fitness: Every potential solution in the population is called a person. Each person has a distinct genetic composition (chromosome), and the quality or efficacy of that solution is reflected in the related fitness score.
Variation: Individuals undergo changes over time through mutations, similar to how living organisms evolve.

In their study, the researchers use the Random Forest (RF) method as a core component of the Genetic Algorithm (GA). They selected RF because it effectively addresses the issue of overfitting, a common challenge with standard Decision Trees. Additionally, RF performs well with various types of data and is particularly useful for datasets with class imbalances. One of the advantages of RF is that it doesn't require special data preprocessing. Alternative tree-based methods such as Extra-Trees and XGBoost can be employed in place of Random Forest (RF).

In their proposed method covers:

Step 1 - Initialization

- Import the cleaned Credit Card Fraud dataset.
- Define variables:
  - List to store feature names.
  - Target variable.
  - Empty array to store optimal feature names.
  - Total number of iterations.

Step 2 - Generate Initial Population

Generate the initial collection of feature labels and store this set within the variable designated as 'A'.

Step 3 - Compute Fitness

For each candidate feature vector (subset of features):

Compute the fitness value q, which determines if the vector is optimal.

If not optimal, perform crossover, mutation, and fitness computation until convergence.

Step 4 - Convergence

The algorithm converges when the maximum accuracy is achieved after k iterations.

With a maximum accuracy rate of 99.98%, the Random Forest (RF) model proved that the recommended method was successful in accurately detecting fraudulent transactions. This performance surpasses previous approaches, demonstrating the benefit of using Genetic Algorithm (GA)-based feature selection to improve the prediction accuracy of machine learning models in identifying credit card fraud.

Bharat Kumar Padhi and colleagues [3] centers on choosing the best features using the Rock Hyrax Swarm Optimization Algorithm. Their study presents a novel feature selection method known as Rock Hyrax Swarm Optimization for Feature Selection (RHSOFS). This method aims to make credit card fraud detection systems work better by finding the most important pieces of information. Traditional feature selection methods frequently struggle to determine the optimal number of features, which may cause the model to either overfit or underfit the data. To address these challenges, RHSOFS employs the RHSO, inspired by the natural behavior of rock hyrax groups in the wild.

The Rock Hyrax Swarm Optimization (RHSO) algorithm takes its ideas from how rock hyraxes naturally act, like how they move around, explore new places, and use resources. These behaviors help the algorithm find the very best group of features. RHSO for Feature Selection method was tested using ten real credit card fraud datasets and its results were compared to several other ways of choosing features; to evaluate its effectiveness, the method was compared with several other optimization-based

feature selection techniques. Its performance was then tested using four different classification approaches.

The experimental results highlight the effectiveness of the RHSOFS approach in identifying the optimal feature subset for credit card fraud detection. When compared to other FSS methods, RHSOFS not only achieved higher classification accuracy but also significantly reduced the number of selected features.

The capacity of RHSOFS to efficiently choose the best feature set, handle enormous amounts of data with many features, and enhance the functionality of credit card fraud detection systems. These benefits make RHSOFS an invaluable tool for academics and experts in credit card fraud detection and other fields where choosing the best characteristics is essential.

In their research, the authors compared RHSOFS against these methods:

- Without Feature Selection (WTFS)
- Genetic Algorithm Feature Selection (GAFS)
- Particle Swarm Optimization Feature Selection (PSOFS)
- Ant Colony Optimization Feature Selection (ACOFS)
- Differential Evolutionary Feature Selection (DEFS)

2.2 Implement Credit Card Fraud Detection

Y. Fang and colleagues [4] used machine learning methods – specifically Random Forest (RF), Gradient Boosting Machine (GBM), and Light Gradient Boosting Machine (LGBM) – Using a dataset containing 410,000 records to address the challenge of detecting credit card fraud. Their paper implements a Light Gradient Boosting Machine (LightGBM) model for spotting fraudulent credit card transactions. Before training the model, they processed the data by using one-hot encoding to deal with any text-based information in certain categories. They also addressed the problem of having many more normal transactions than fraudulent ones by using the Smote algorithm. The researchers then checked how well the LightGBM model performed and compared it to commonly used methods like GBM and RF. Their findings showed that the LightGBM model was better in terms of remembering all the fraudulent cases (Recall rate) and also took less time to train. To further show that the model could work well on new, unseen data, they tested it on another real-world dataset and found that LightGBM continued to perform strongly. In summary, According to the authors, the types of features utilized in their method are outlined as follows:

1. Age Analysis

Customers over 50 are scrutinized, as they typically have credit cards due to substantial assets. Age distribution is categorized into intervals (0-7) for analysis.

2. Zip Code Evaluation

Zip codes are examined to identify high-risk areas for credit card fraud. Emphasis on frequently occurring zip codes based on transaction volume and amounts.

3. Transaction Amount Significance

Transaction amounts are considered crucial, with sudden increases signaling potential fraud.

4. Consumption Category Insights

Sixteen merchant categories are analyzed to understand customer spending habits. Identification of unusual purchases, such as cars, aids in fraud detection.

5. Job Category Stability

Stable professions, like civil servants or teachers, are perceived as less likely to engage in credit card fraud.

6. Phone Number Monitoring

Phone numbers linked to credit cards are monitored for suspicious activity. High-frequency credit card spending messages within 24 hours suggest potential fraud.

LightGBM achieves the highest value of 99%. This high rating shows how well LightGBM distinguishes between authentic and fraudulent transactions.

Related research was conducted by S P Maniraj and associates [1] utilizing the Isolation Forest and Local Outlier Factor (LOF) algorithms. Their goal was to investigate how data science and machine learning methods might be used to detect credit card fraud. The study emphasizes how important it is to identify fraudulent transactions in order to safeguard clients and reduce monetary losses. They proposed a model of previous credit card transactions and then look for any suspicious patterns using algorithms that detect anomalous activity.

The study is based on two algorithms, Local Outlier Factor (LOF) and Isolation Forest, to analyze a huge pre-requited dataset of card transactions. The LOF algorithm measures the local deviation of data points from their neighbors, while the Isolation Forest algorithm isolates observations by randomly partitioning the data.

The study presents the results of applying these algorithms to real credit card transaction data, showing that both approaches were effective in detecting fraudulent activity, but the Isolation Forest algorithm was more precise than the Local Outlier Factor (LOF).

The authors highlight the practical challenges and limitations of deploying such a system in real-world scenarios, stressing the importance of collaboration with banks and the need to ensure privacy and confidentiality of data. They also note that the results may differ based on the dataset used and the selected parameters for the anomaly detection algorithms.

Sarthak Aggarwal and colleagues [5] investigated the use of machine learning (ML) techniques, including Support Vector Machine (SVM), Logistic Regression, Decision Tree, and Random Forest, to detect credit card fraud. The first section of the article discusses the pervasive problem of credit card fraud in the contemporary digital age and highlights the need for effective detection techniques to counter it.

The paper then goes into detail about several machine learning methods that are often used for fraud detection, including:

1. Support Vector Machine (SVM)

It segregates data using hyperplanes with dimensions dependent on features, classifying data points close to the hyperplane.

2. Logistic Regression

It's suited for scenarios with categorical dependent variables and evaluates transaction characteristics to assess their legitimacy.

3. Decision Tree

It recursively divides datasets to create tree-like structures, with internal nodes for attribute tests and leaf nodes holding class labels.

4. Random Forest

This method creates several decision-making structures and combines their results to achieve better accuracy than using single structures alone.

This study looking at how well they perform in terms of correctly identifying fraud (precision), overall correctness (accuracy), and how often they wrongly flag normal transactions (false alarm rate). This was done using real data containing both genuine and fraudulent transactions. The findings show that the Decision Tree method performed best, achieving a high accuracy of 99.94%, correctly identifying 99.99% of the fraud, and having a very low false alarm rate of 0.042. Logistic Regression was also quite good, with an accuracy of 99.92%, correctly finding 99.98% of the fraud, and a false alarm rate of 0.063. This points out how effective machine learning is for detecting credit card fraud, with the Decision Tree method standing out as particularly strong. These results could help in creating more dependable fraud detection systems to protect both customers and banks.

K. Madkaikar and colleagues [6] assessed and compared how well seven different machine learning methods worked for finding credit card fraud. These methods were LR, NB, RF, KNN, GB, SVM, and NN. Among these, the Gradient Boosting method performed best.

In order to detect fraudulent credit card activity, F. K. Alfaraj and colleagues [7] published a unique method that makes use of state-of-the-art deep learning algorithms. Their study used real credit card transaction data to assess the efficacy of many

machine learning and deep learning models. Their convolutional neural network (CNN) model achieved 99.9% accuracy, an F1-score of 85.71%, a precision of 93%, and an AUC score of 98%, which was significantly superior than other methods. These results show how successfully deep learning can detect fraud and provide researchers and industry experts with important new information.

In their evaluation of previous studies on credit card fraud detection that used machine learning and deep learning techniques, the researchers discovered issues including inconsistent data, difficult feature selection, and the requirement for more sophisticated models. To automatically extract pertinent information from the transaction data and classify transactions as either real or fraudulent, they put in place a multi-layered convolutional neural network (CNN). This model was tested and trained using a dataset of real credit card transactions. After analyzing past research on credit card fraud detection using machine learning and deep learning methods, the researchers found problems such imbalanced data, challenging feature selection, and the requirement for more advanced models. A multi-layered convolutional neural network (CNN) was implemented to automatically extract relevant elements from the transaction data and identify transactions as either fraudulent or legitimate. To train and evaluate this model, an actual dataset of credit card transactions was utilized.

For their study, the authors employed the European card benchmark dataset for credit card fraud detection, which encompasses over 280,000 transactions, with only a small fraction representing fraudulent activity. They performed preliminary steps like selecting relevant features and normalizing the data to prepare it for the deep learning model.

Their study's CNN model produced impressive results with an accuracy of 99.9%, an F1-score of 85.71%, a precision of 93%, and an AUC score of 98%. The CNN model fared better on every metric than a number of machine learning algorithms, such as XGBoost, Decision Tree, Random Forest, Support Vector Machine (SVM), Logistic Regression, Extreme Learning Machine (ELM), and others.

The authors established new standards for dataset performance by effectively demonstrating the capacity of their CNN model to identify credit card fraud. These findings highlight the enormous potential of deep learning algorithms in the field of fraud detection and provide both academics and industry practitioners with helpful information.

The study's authors demonstrated the significant effectiveness of their CNN model in identifying credit card fraud, establishing new performance benchmarks on the data examined. These results underscore the substantial promise of deep learning algorithms for fraud detection and offer significant knowledge for academics and experts in this area.

Based on the evaluation above, the highest accuracy achieved with the financial dataset is listed below:

1. Random Forest

In their paper, V. Sahaya Sakila and colleagues [8] offer a thorough explanation of the Random Forest algorithm. They describe it as a method for classifying data that builds numerous decision-making structures using different parts of a given dataset. Instead of depending on just one decision structure, the algorithm combines the results from all of them, deciding the final outcome by looking at the most common prediction among these structures. This combined approach improves how accurate the model is.

The graphic below illustrates the Random Forest algorithm's functioning and the decision-making process of many decision trees. As seen in Figure 2.4, the data is first separated into two parts: one for model training and another for performance evaluation. The model produces smaller, more varied data sets—sometimes including the same data point more than once—by randomly choosing samples throughout the training phase. These smaller sets are then used to construct different decision trees, each of which generates straightforward predictions. After all the decision trees have been trained, their forecasts are combined. For classification challenges, the category that receives the most votes from the trees is selected as the final answer. The average of each individual tree forecast is used for regression tasks. This approach results in reduced error rates and increased accuracy.

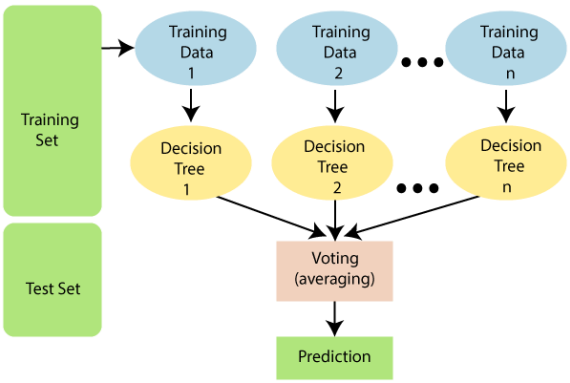

Figure 2.2

behavior of random forest algorithm. (Source:[8]*)*

2. Logistic Regression

A common machine learning technique for classifying data into two groups is logistic regression, according to J. Ren and colleagues [22]. It gives the likelihood that a data piece belongs to a particular category a value between 0 and 1. In contrast to linear regression, logistic regression does not assume that the input and output variables have a linear relationship. In order to model the probability of a yes/no outcome without requiring a straight linear link, it employs a nonlinear logarithmic transformation on

the odds ratio, which allows it to comprehend more complex relationships. This adaptability makes logistic regression especially helpful when the relationship between the factors predicting the outcome and the logarithm of the odds of that outcome is not strictly a straight line, which can make linear regression less appropriate in some cases. As illustrated in Figure 2.5, when input values ranging from -20 to 20 are used in the logistic function equation, where 'x' represents the input, the resulting outputs are scaled to a range between 0 and 1. This demonstrates how the sigmoid (logistic) function compresses the input values, allowing logistic regression to effectively model probabilities for tasks involving the classification of data into two categories.

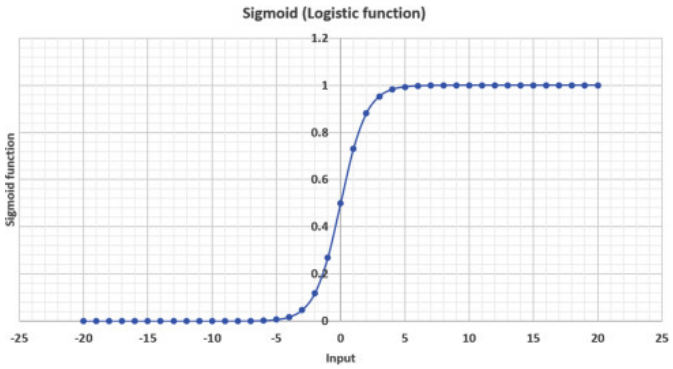

Figure 2.3

behavior of logistic regression algorithm. (Source: [22]*)*

3. Support Vector Machine (SVM)

J. Ren [22] describes Support Vector Machine (SVM) is a supervised machine learning algorithm use for classifying and predicting outcomes. It excels in situations where a distinct separation between various classes is essential, especially in spaces with numerous dimensions. SVM's primary goal is to identify an optimal boundary, often referred to as a hyperplane, that effectively distinguishes between different groups of data points. This makes SVM particularly suitable for tasks such as categorizing items or predicting values when faced with complex and multi-dimensional datasets.

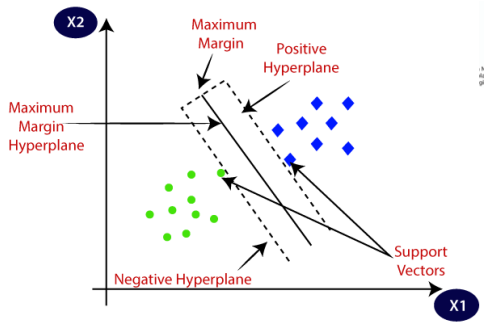

Figure 2.4

behavior of SVM algorithm (Source: [22]*)*

In Figure 2.6 Support Vector Machines (SVM), there are two key terms that will be frequently reiterated:

Support Vectors - Support Vectors are specific data points that play a crucial role in determining the hyperplane. These are the points closest to the hyperplane and are instrumental in defining the separating line in SVM.

Margin - The distance between the dividing line (hyperplane) and the support vectors—the data points closest to it is referred to as the margin. A larger margin is preferable in Support Vector Machines (SVM) since it indicates a clearer separation between the various categories. The next sections will provide a more thorough explanation of the two types of margins: the hard margin and the soft margin .

4. Decision Tree

According to J. Ren and colleagues [22], a decision tree is similar to a flowchart diagram, as shown in Figure 2.7, where the end boxes (leaf nodes) indicate the final conclusions, the boxes inside represent features, and the lines linking them reflect decision rules. This technique is a useful tool in supervised machine learning since it is highly adaptable and can be used for both categorising data and predicting numerical values. The creation of Random Forest, a powerful method that combines many decision trees trained on different data segments, also relies heavily on decision trees. This combo improves Random Forest's machine learning performance and popularity.

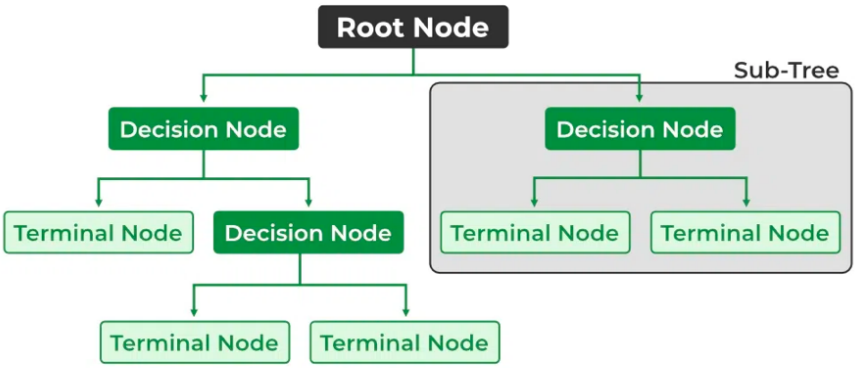

Figure 2.5

behavior of decision tree algorithm (Source: [22])

2.3 Adversarial Attacks in Adversarial Environment

V. Sahaya Sakila and colleagues [8] clarify in their paper that an adversarial attack involves changing input data in a way that tricks a machine learning model into making wrong guesses. This trickery involves making small changes to the input data that people wouldn't notice but are enough to confuse the model. For instance, an adversarial attack could slightly change the colors in a picture, causing a system that

recognizes images to incorrectly identify the object. Such misleading inputs can have serious consequences, like making a self-driving car misread road signs or a face recognition system make wrong identifications. Essentially, adversarial attacks take advantage of weak points in machine learning models, showing that they can be fooled by small changes to what they see.

Anant Jain [10] discusses in his paper that adversarial examples are inputs created specifically to make machine learning models give incorrect predictions. These examples are often used to attack deep neural networks, which, despite being powerful, are very vulnerable to such attacks. Adversarial examples can be made by adding a little bit of "noise" to a picture or by using more complex methods. Even though this noise is often something humans can't see, it can greatly affect how accurate the model is.

Even though there are several ways to defend against adversarial attacks, none of them work perfectly all the time. The best defense is to use a mix of different techniques that make it harder for attackers to create these misleading examples. Adversarial examples are a big problem for machine learning, but they also give researchers valuable information, helping them understand the limits of current models and encouraging the development of stronger solutions.

Han Xu and colleagues [9] offer a thorough investigation into adversarial attacks and defenses in their research, with a particular focus on deep neural networks (DNNs). They explore the idea of adversarial examples, which are specially created inputs intended to trick DNN models. The authors discuss different kinds of attacks, including those that aim to evade detection and those that aim to corrupt the training data, as well as attacks where the attacker knows the model's details (white-box) and those where they don't (black-box). They also review various methods developed to protect against these attacks, such as hiding the model's gradients, training the model with adversarial examples, and mathematically proven defenses. Furthermore, their analysis extends to adversarial attacks and defenses in data that isn't images, specifically data organized as graphs. They introduce DeepRobust, a software library built using PyTorch for studying adversarial learning, and finish by pointing out current challenges and unsolved problems in this area.

The lessons they offered is well-structured and covers a wide range of subjects pertaining to adversarial assaults and defenses. It is intended for researchers, college students, and professionals interested in machine learning and deep learning. The presenters have made important contributions to research on this subject and are acknowledged authorities in the field. A practical example of the DeepRobust library is included in the tutorial, making it a helpful resource for practitioners and researchers in this area.

In summary, Figure 2.9 provides a complete and current overview of adversarial attacks and defenses, making it a very valuable resource for anyone interested in this quickly evolving area of research.

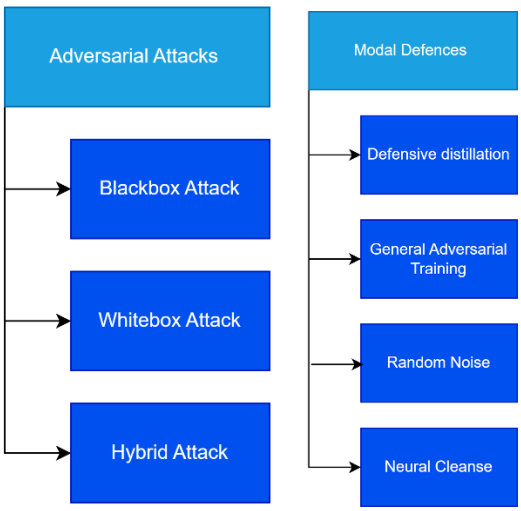

Figure 2.6

overview of adversarial attacks and defense mechanisms (Source: [9]*)*

F. K. Alarfaj et al. [7] describe an adversarial attack as an intentional manipulation of input data to deceive a machine learning system, causing it to make wrong decisions or predictions. These attacks are particularly common in AI models, such as those used in recognizing images, processing language, or controlling autonomous systems. The altered input is known as an adversarial example. Even though the changes to the input might be so small that a human wouldn't notice, these subtle tweaks can easily trick AI models into making mistakes.

The objective of an adversarial assault, as stated in equation 2.1, is to identify a little alteration, denoted by δ, that, when introduced to an initial input x, causes a neural network model f(x) to incorrectly categorize that input. The attacker attempts to resolve this issue mathematically.

$$\delta_{\min} L(f(x+\delta), y) \tag{2.1}$$

Where:

- x represents the original input (e.g., an image).
- f(x) refers to the model's prediction (e.g., classification scores).
- L(·) represents the loss function, which is a way to measure how far off the model's prediction is from the actual correct category.
- y is the actual label of x.

The adversarial perturbation δ should be small, typically constrained by some norm.

The attacker's sole option is to engage with the model by posing queries and observing its responses in order to determine how to provide deceptive instances. The goal is to alter input x so that the model interprets it incorrectly. This assault may be explained mathematically in this way [14].

I. Objective of Attack

The attacker aims to identify a small perturbation δ that causes the model to $f(x)$, which correctly classifies input x, misclassifies the perturbed input $x_{adv} = x + \delta$.

Mathematically, the goal is to maximize the loss function $L(f(x_{adv}), y)$ where $f(x_{adv})$ is the model's output for the perturbed insert and y is the true label of x.

The goal is represented in equation 2.2,

$$x_{adv} = x + \delta \; such \; that \; f(x_{adv}) \neq y \tag{2.2}$$

II. Loss Function of Attacks

The typical loss function for untargeted attacks (where any wrong prediction is acceptable) is,

$$L(f(x), y) = - \; log \; P(y \mid x) \tag{2.3}$$

Where:

- $P(y \mid x)$ represents the likelihood that the model predicts the correct label for the given input.x.

In the case of a targeted attack, where the attacker intends for the model to predict a specific incorrect class y_{target}, the loss function becomes as equation 2.4:

$$L(f(x), y_{target}) = -logP(y_{target} \mid x) \tag{2.4}$$

Considering adversarial attacks can be divided into 3 parts as Whitebox, Blackbox and highbred attacks.

1. Whitebox Attack

M. Zhou et al. [23] describe a white-box attack in machine learning as an attack where the adversary possesses full access to the model's structure, parameters, and gradients. With this detailed insight, the attacker can create powerful adversarial examples that are more likely to successfully mislead the model.

Characteristics of a white-box attack include:

- The attacker takes complete access to the model's methods, parameters, and gradients.
- This also includes knowledge of the training dataset, model weights, and potentially the loss function used.
- The attacker leverages detailed information to design and execute targeted attacks more efficiently.

- Since the attacker can accurately calculate the gradients and fine-tune the input systematically, the attacks are able to be more accurate and better optimized.
- Generally faster and more efficient due to direct access to gradients.
- Can often achieve lower perturbation levels to mislead the model.
- Useful for evaluating the robustness of a model when the attacker is assumed to have insider knowledge (e.g., during testing or training phases).

Examples for Whitebox attack:

Fast Gradient Sign Method (FGSM), Projected Gradient Descent (PGD), Carlini & Wagner Attack, Deep-Fool, Universal Adversarial Perturbations

White-box attacks leverage detailed knowledge of a model's architecture and parameters to generate highly effective adversarial examples. This ability allows for testing the model's resilience and identifying potential weaknesses, which can then be addressed by implementing defenses to protect against these vulnerabilities.

2. Blackbox Attack

S. Kaviani and colleagues [24] explain that a black-box attack is a way to attack a model where the attacker cannot see how the model is built, its internal settings, or how it learns. Instead, the attacker can only give the model data and see what it outputs. This situation is similar to real-world uses, like when a model is part of an online service (API) and the user can't see how it works or what it knows.

Characteristics of Blackbox attack,

- The internal structure of the model, for instance, parameters and gradients are completely unknown to the attacker.
- This interaction with the model can only be done through input-output queries.
- Must rely on approximations, such as finite difference methods, to estimate gradients.
- This typically involves multiple queries to the model in order to make an inference about its behavior, hence increasing computation through this process.
- Typically, much slower and more resource-intensive, given the need for multiple queries and gradient estimations.
- The attacker might need to use more significant perturbations because they lack detailed knowledge about how the model reacts.
- This type of attack is relevant in real-world scenarios, as models are often deployed in environments—such as cloud services or APIs—where attackers do not have direct access to the system.
- Simulates an adversary with limited information, which is common in practice.

Examples for Blackbox attack:

Zeroth-Order Optimization (ZOO), Transfer Attacks, Query-Based Attacks, Adversarial Patch, Black-Box Decision Boundary Attacks

Black-box attacks show that even without insider information, adversaries can successfully attack machine learning models. These are also attacks that highlight robust models, key concerns when models are deployed to real environments where direct access to model parameters is not feasible.

Comparison of Whitebox and Blackbox attacks describe in table 2.1,

Table 2.1

Detailed description of adversarial attack types

Feature	**White-Box Attack**	**Black-Box Attack**
Model Access	Complete (weights, gradients)	None (only inputs/ outputs)
Gradient Use	Directly uses gradients	Estimates gradients from outputs
Efficiency	More efficient, faster	Generally slower, more queries
Attack Precision	High precision	Lower precision due to estimation
Common Techniques	FGSM, PGD, Deep-Fool	Zeroth-order optimization, transfer attacks
Real-World Relevance	Less common in real scenarios	Common in deployed systems

3. Hybrid Attack

S. Kaviani et al. [24] describe a hybrid attack as a strategy that combines elements of both white-box and black-box attacks to create more effective adversarial examples or exploit weaknesses in machine learning models. This approach unfolds in three phases:

a) Initial Black-Box Phase: The attacker starts by treating the model as a black box, interacting with it through various inputs to observe its behavior and decision boundaries without accessing its internal architecture or parameters. During this phase, the attacker collects output data for specific inputs to estimate the model's decision boundaries.

b) Model Approximation Phase: The attacker creates a surrogate model that replicates the original model's decision function using the data gathered during the black-box phase. Either labelled data or the outputs from the original model are used to train this surrogate model. Sing the information collected during the black-box phase, the attacker builds a surrogate model that mimics the decision function of the original

model. This surrogate model is trained with either labeled data or the outputs obtained from the original model.

c) White-Box Phase: After building this stand-in model, the attacker can then perform attacks where they have full knowledge of this substitute. At this point, the attacker can see how the substitute model learns and its settings, allowing them to use methods like FGSM or PGD. The misleading examples created using the substitute model are then tested on the original, real model.

Examples for Blackbox attack:

Elastic Net, Substitute Model Attack, Partial Knowledge Attack, Query-Efficient Black-Box Attack, Model Inversion Attack, Hybrid Evasion and Poisoning Attack

2.3.1 Zeroth Order Optimization

J Goodfellow et al. [15] describe the Zeroth-Order Optimization (ZOO) attack as a black-box strategy in adversarial machine learning. This approach allows attackers to generate adversarial inputs—designed to deceive machine learning models—without needing access to the model's internal parameters or gradients. Instead, ZOO uses zeroth-order optimization, which approximates the gradient by relying solely on observing the model's outputs in response to specific inputs. As a result, the attacker can manipulate the model by interacting with its outputs, but lacks insight into the model's internal workings.

The ZOO attack is based on finite difference methods, where the attacker approximates the gradient by making small perturbations to the input and observing how the model's output changes. This technique allows the attacker to optimize the input even without knowing the true gradient of the model.

Example of ZOO Adversarial Attack,

1. Begin with an original image, x, that the model correctly classifies.
2. Select a target class (or simply aim to misclassify the image).
3. Gradually estimate the gradient of the model's loss function by applying finite difference methods to each component of the input
4. Apply small adjustments in the direction of the estimated gradient to gradually shift the input closer to the decision boundary of the target class.
5. Stop when the model misclassifies the input, creating an adversarial example.

Key Components of the ZOO Attack,

1. Objective - The attacker aims to generate an adversarial example $x_{adv} = x + \delta$ such that the model misclassifies it. The attacker does this by maximizing the model's loss function $L(f(x), y)$ with respect to the input x, where y is the true label of x.

2. Gradient Approximation - Since the attacker doesn't have access to the true gradient $\nabla_x L(f(x), y)$, the ZOO attack approximates it using finite differences.

Mathematical Formulation of the ZOO Attack,

1. Finite Difference Gradient Estimation:

The gradient loss function $L(f(x), y)$ with respect to the input x_i (the i-th component of the input) is estimated to use the following finite difference approximation as equation 2.5,

$$\frac{\partial L(f(x), y)}{\delta x_i} \approx \frac{L(f(x + \delta e_i), y) - L(f(x), y)}{\delta} \quad (2.5)$$

Where:

- δ is a small perturbation applied to the input.
- e_i is the standard basis vector (a vector where all components are zero except for the i-th component, which is 1).
- $L(f(x + \delta ei), y)$ is the loss function evaluated at the perturbed input $x + \delta e_i$
- $L(f(x), y)$ is the loss function evaluated at the original input x.

This method requires two queries to the model for each dimension x_i, one for the original input x and one for the slightly perturbed input $x + \delta e_i$.

2. Gradient-Based Optimization:

Once the gradient is approximated using the finite difference method, the attacker uses it to update the input x to generate the adversarial example. The update rule is like first-order methods but uses the estimated gradient as equation 2.6.

$$x_{adv} = x + \eta \cdot sign(\hat{g}) \quad (2.6)$$

Where:

- η is the learning rate or step size.
- $\hat{g}$ is the estimated gradient vector, with each component $\hat{g}\imath$ computed using the finite difference method.
- $sign(\hat{g})$ It is the sign of the estimated gradient that indicates the direction in which each component of the input should be altered.

The process is repeated iteratively to refine the adversarial example.

Key characteristics of ZOO-based attacks,

1. Black-box nature - ZOO attacks assume that the attacker does not have access to the model's architecture, parameters, or gradients. The only thing available is the model's output (e.g., classification scores).

2. Gradient estimation - Since the attacker cannot compute the gradient directly (as would be done in a white-box attack like FGSM or PGD), ZOO approximates the

gradient using finite differences. Essentially, the attacker perturbs the input slightly and observes the change in the model's output to estimate how the model behaves.

3. Optimization step – After approximating the gradient, traditional optimization methods like gradient descent are used to create the adversarial example. These methods help find a perturbation that alters the input, causing the model to misclassify it.
4. Query-dependent – ZOO attacks typically need numerous model queries to precisely estimate the gradient, making them more resource-heavy than white-box attacks.

How ZOO works,

1. Input modification – The attacker begins with an original input (like an image) and introduces a small noise to perturb it.
2. Model interaction – The attacker sends the modified input to the model and monitors the output (e.g., predicted class or probabilities).
3. Gradient approximation – By observing how the model's output shifts with slight input changes, the attacker estimates the gradient.
4. Iterative refinement – With the estimated gradient, the attacker adjusts the perturbation step by step to increase the likelihood that the model will misclassify the input.
5. Adversarial example creation – After sufficient iterations, the modified input becomes an adversarial example that the model misinterprets.

Applications of ZOO Attacks,

1. Black-box attack scenarios - ZOO attacks are useful when attackers don't have access to the model's internal structure, such as in commercial systems (e.g., cloud-based AI services or APIs).
2. Valuating resilience – Researchers can use ZOO attacks to assess how well machine learning models can withstand adversarial attacks.

Drawbacks of ZOO Attacks:

1. High query cost - Since ZOO relies on querying the model multiple times to estimate gradients, it requires many queries, making it impractical for some real-world applications.
2. Slower compared to white-box attacks - The reliance on gradient approximation makes ZOO much slower than white-box attacks, which can directly access the gradient.

ZOO adversarial attacks are a potent black-box approach for generating adversarial examples when the internal details of a machine learning model are unavailable. However, they necessitate considerable computational power and numerous queries.

2.3.2 Deep Fool adversarial attack

J. Lin et al. [13] describe the Deep Fool adversarial attack as a white-box technique focused on generating adversarial examples by determining the smallest perturbation required to mislead a machine learning model, particularly deep neural networks (DNNs). The aim of Deep Fool is to adjust the input just enough to cross the model's decision boundary, causing misclassification, while keeping the perturbation minimal and typically undetectable by humans. The algorithm works iteratively, refining the input to gradually push it toward the decision boundary of an alternative class, with the goal of finding the smallest change needed to alter the model's prediction.

Step-by-Step Mathematical Implementation,

Let $f(x)$ represent the classifier's output for input x, where x is a vector representing an input sample (e.g., an image). The classifier outputs the class with the maximum score from $f(x)$.

1. Initial Setup

Given an input image x_0, seek the minimal perturbation δ such that as equation 2.7,

$$\hat{k} = arg\ max\ f(x_0 + \delta)\ \neq arg\ max\ f(x_0) \tag{2.7}$$

2. Linear Approximation of the Classifier

To simplify the problem, DeepFool uses a first-order Taylor approximation to linearize the classifier around the current input x. For binary classification, the decision boundary is approximated by,

$$f(x) \approx f(x_0) + \nabla f(x_0)\top(x - x_0) \tag{2.8}$$

where $\nabla f(x_0)$ presents the gradient of the classifier's output in relation to the input a x_0.

3. Computing the Perturbation

The goal is to find the smallest perturbation that brings x_0 to the decision boundary. For a binary classifier, the minimal perturbation can be computed as equation 2.9,

$$\delta = -\frac{f(x_0)}{\|\nabla f(x_0)\|_2^2}\nabla f(x_0) \tag{2.9}$$

where $\|\cdot\|_2$ denotes the L2 norm.

4. Extension to Multiclass Classification

For multiclass classification, the process is repeated iteratively. The algorithm identifies the closest decision boundary for a different class and computes the perturbation that would push x across that boundary. At each iteration 3rd step, the perturbation δ_i can be updated as equation 2.10,

$$\delta_i = \min_{j \neq k} \frac{| f_j(x_i) - f\,\hat{k}(x_i) |}{\| \nabla f j(xi) - \nabla f\,\hat{k}(x_i) \|_2} \quad (2.10)$$

where $f_j(x_i)$ represents the classifier's output for class j at iteration i, and $\nabla f_j(x_i)$ is the gradient of the classifier for class j.

5. Iteration Until Misclassification

The algorithm continues updating the perturbation δ and x, iteratively moving closer to the decision boundary, until $x + \delta$ is misclassified.

Example for DeepFool attack type,

DeepFool achieves this by progressively determining the direction to adjust the input, using a linear approximation of the model's decision boundaries. Even with minimal alterations, this attack can result in misclassifications, demonstrating the susceptibility of deep learning models to slight modifications.

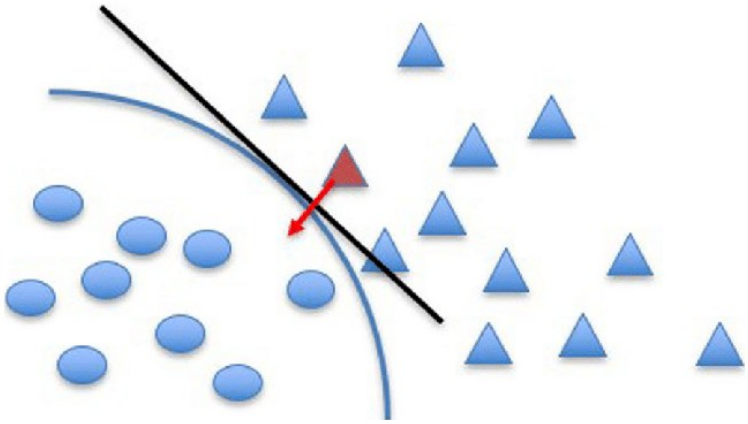

Figure 2.7

behavior of DeepFool attack (Source: [13])

As figure 2.10 works by drawing a straight line to estimate the decision boundary of a deep learning model near a specific input (like a triangle). It then looks for the shortest path to slightly change the input so that it crosses this boundary, causing the model to make a targeted misclassification. This approach aims to find the quickest and smallest modification needed to deceive the model.

Chen et al. [17] explain the DeepFool attack as an iterative adversarial technique designed to identify the smallest perturbation that causes misclassification. It uses linear approximations of the decision boundaries to efficiently compute the direction and size of the required adjustments. This attack can be applied in both targeted and untargeted versions, achieving a high success rate across different deep learning models. While it is computationally efficient and effective at finding minimal perturbations, it may struggle against models with robust adversarial defenses.

Additionally, the attack's effectiveness can be impacted by the dimensionality of the input data.

2.3.3 Elastic Net adversarial attack

Chen et al. [18] introduce the Elastic Net Attack on Deep Neural Networks (EAD), a type of adversarial attack that creates adversarial examples by merging elements of both L1 and L2 norm-based attacks. The Elastic Net approach uses an elastic-net regularization term, which combines the sparsity-promoting L1 norm and the smoothness-enhancing L2 norm, to identify perturbations that can successfully deceive a deep learning model.

Characteristics of the Elastic Net Attack,

1. Combines L_1 and L_2 Norms**,**

EAD combines L1 and L2 regularization terms in its objective function to regulate both the sparsity and the size of the perturbation. This enables the algorithm to produce perturbations that are sparse (affecting only a few pixels) and small in magnitude.

2. Generalizes the Carlini & Wagner (C&W) Attack

EAD extends the well-known Carlini & Wagner L_2- based attack by adding an L_1 norm term. When the coefficient for the L_1 norm is set to zero, EAD reduces to the original C&W L_2 attack.

3. Optimization Problem

The objective function of EAD seeks to reduce the difference between the original input and the adversarial example, while also increasing the classification loss to ensure misclassification. The combined regularization term promotes a perturbation that is both sparse and minimal in size represented in equation 2.11,

$$minimize\ c \cdot f(x+\delta) + \beta \parallel \delta \parallel 1 + \parallel \delta \parallel_2^2 \tag{2.11}$$

where $f(x+\delta)$ is the loss for the perturbed value, δ is perturbation, c is a constant to balance the classification loss and the perturbation size, and β is a parameter controlling the L_1 regularization.

4. Adversarial Example Generation

EAD iteratively optimizes the above objective function to generate an adversarial example. The optimization seeks to find a perturbation δ such that the modified input $x+\delta$ is classified as a different class, while keeping the perturbation as small and sparse as possible.

5. Effectiveness

EAD has proven to be successful in producing adversarial examples for convolutional neural networks (CNNs) and other deep learning models. It sometimes outperforms other attack techniques, especially when the sparsity-promoting aspect of the L1 norm contributes to the creation of subtle and difficult-to-detect adversarial samples.

The Elastic Net adversarial attack (EAD) generates adversarial examples by optimizing an objective function that includes both L_1 and L_2 norms, making it a generalization of the Carlini & Wagner attack. It balances sparsity and smoothness in the perturbation to effectively fool deep learning models, especially in scenarios where sparse perturbations are more effective.

2.4 Robustness Predictions in Adversarial Environment

The Global Risk Institute [21] reports that financial institutions have been slow to implement defenses against adversarial attacks, with 25 out of 28 financial organizations not taking any measures to address these threats. As a result, to explore defense strategies, I expanded my research beyond the financial sector. In this context, V. Sahaya Sakila et al. [8] introduced an innovative approach to improving classifier robustness against adversarial examples using a Generative Adversarial Network (GAN) framework. Their method, called Generative Adversarial Trainer (GAT), features a generator that identifies vulnerabilities in a classifier and uses these weaknesses to create perturbations. The classifier then reverts the generated image to its original label, creating a cycle of learning between the generator and classifier. This process ultimately strengthens the classifier's resistance to adversarial images, making it harder for the generator to deceive the model. A key advantage of this approach is its practicality, as it does not require expensive optimization steps to find optimal adversarial images. Additionally, the classifier trained with this method becomes highly resilient to adversarial examples and serves as a form of regularization for neural networks.

2.4.1 Defensive Distillation

Defensive distillation is a strategy developed to protect machine learning models, particularly deep neural networks, from adversarial attacks, according to Papernot and colleagues [19]. Distillation was originally intended to reduce the size of models, but it was later modified to increase a model's resilience to these attacks. By reducing the model's sensitivity to slight modifications in the input data, the technique makes it more difficult for adversarial samples to result in inaccurate classifications.

How Defensive Distillation Works,

1. Standard Distillation

In the standard way of doing distillation, a "teacher" model is first trained using the original data. Then, the "soft labels" from this teacher model which are the probabilities it assigns to different possible categories – are used to train a "student" model. The student model learns from these probability distributions instead of just

the original, definite category labels. This process helps to pass on knowledge from the teacher to the student and can make the model smaller.

2. Temperature Scaling

In defensive distillation, the training process introduces a "temperature" parameter T during the distillation phase. The softmax output of the model is modified to generate softened probability distributions over the classes as equation 2.12:

$$softmax(z_i/T) = \frac{exp(zi/T)}{\sum jexp(zj/T)} \quad (2.12)$$

where z_i is the logit for class i, and T is the temperature. Higher values of T produce softer probability distributions, making the output less confident.

During training, the model is trained with a high temperature to produce smooth class probabilities. When the training is finished, the temperature is set back to 1 for normal prediction.

3. Training the Student Model with Soft Labels

The softer output probabilities generated by the teacher model at a higher temperature are used to train the student model. This method lessens the impact of adversarial perturbations and increases the student model's resilience to attacks by making it less sensitive to slight changes in the input.

Why Defensive Distillation Works,

1. Smoothing Decision Boundaries

By using soft labels, the decision boundaries learned by the distilled model are smoother and less sharp than those learned with traditional training. This makes it harder for small adversarial perturbations to push an input across a decision boundary, thereby reducing the effectiveness of adversarial attacks.

2. Reducing Gradient Magnitude

In order to make it more difficult for gradient-based attacks, such as the Fast Gradient Sign Method, to identify the best paths for input manipulation, defensive distillation might lessen the size of the model's gradients.

Limitations and Challenges,

1. Adaptive Attacks

While defensive distillation initially showed promising results against some adversarial attacks, later research demonstrated that adaptive adversarial attacks could

still bypass the defense. Attackers could adjust their strategies to account for the modified gradients or directly target the distilled model.

2. Not a Complete Defense

While defensive distillation can lower the success rate of certain attacks, it is not guaranteed defense. Attackers can still create adversarial examples that bypass distilled models, particularly with more sophisticated methods.

3. Increased Training Complexity

The distillation method introduces an additional layer to the training process, involving both a teacher and a student model. This added complexity results in higher computational demands and increases the overall cost of training the model.

Defensive distillation is a technique that uses altered probability scores to reduce the sharpness of deep learning models' decision-making, hence increasing their resistance to adversarial attacks. This method lessens the model's sensitivity to little variations in the input data. Even though it offers some defense, more skilled attackers can still develop ways to circumvent this defense, therefore it's not a complete answer.

2.4.2 General Adversarial Training

General Adversarial Training is a defensive technique presented by J. Goodfellow and colleagues [15] that aims to fortify machine learning models particularly deep neural networks against adversarial challenges. Using this technique, conventional samples are included to the training data together with hostile instances. This enables the model to develop the ability to identify and defend against detrimental input changes. Enhancing the model's capacity to handle the little, targeted changes frequently employed in adversarial attacks is the main objective.

How General Adversarial Training Works,

1. Generating Adversarial Examples During Training

The model's current performance is used to produce false samples at each training phase. These instances are challenging for the model to accurately classify since they are created by making minor, intentional modifications to the original training data. The Carlini & Wagner (C&W) approach, the Projected Gradient Descent (PGD), and the Fast Gradient Sign Method (FGSM) are commonly used to construct these deceptive instances. Every step of the training process generates bogus instances based on the model's current performance. Since these examples are produced by making small, deliberate changes to the initial training data, they are difficult for the model to correctly categorize. These misleading examples are frequently created using the Carlini & Wagner (C&W) attack, the Projected Gradient Descent (PGD), and the Fast Gradient Sign Method (FGSM).

2. Combining Adversarial and Clean Data

Both the original, unaltered instances and the deceptive, hostile ones are included to the training data. After that, the model is trained to accurately classify both kinds of input. Reducing the error for both the adversarial and regular cases is the aim during training, which strengthens the model's resistance to deliberate, minor input modifications.

3. Iterative Process

Creating false instances and then training the model again are recurring steps in an ongoing cycle. As training progresses, this continuous process enables the model to gradually adjust to increasingly potent attacks.

Adversarial examples are continuously updated based on the model's current state, helping the model learn to resist different types of perturbations.

Why General Adversarial Training Works,

1. Increases Model Robustness

Training the model with adversarial examples helps it learn to properly classify altered inputs. As a result, the model becomes more resilient to future attacks, as it becomes accustomed to the types of perturbations typically used in these attacks.

2. Makes Decision Boundaries Smoother

Adversarial training aids in reducing the model's decision bounds. This lessens the effectiveness of attacks that take advantage of sharp decision borders by making it more difficult for minor perturbations to transfer an input from one class to another.

3. Adaptable to Different Attacks

The approach can be adapted to various adversarial attack methods. By training with stronger or different types of attacks, the model can become more resistant to a wide range of adversarial techniques.

Challenges and Limitations,

1. Higher Computing Requirements

Creating and using adversarial examples during training increases the total computing resources needed. This method can take a significant amount of time, especially with large amounts of data and complicated models.

2. Limited Defense Against Adaptive Attacks

While adversarial training improves robustness against known attack methods, adaptive adversaries can still find ways to craft successful adversarial examples that bypass the defense.

3. Overfitting to Specific Attacks

If a model is trained solely on adversarial examples generated with a particular technique (e.g., FGSM), it may become more susceptible to other attack types. To

mitigate this, it is important to incorporate a variety of adversarial example generation methods during training.

Furthermore, Cohen et al. [20] discuss General Adversarial Training as a defense strategy that improves the resilience of machine learning models by exposing them to both adversarial and clean data. This method helps the model better handle adversarial attacks by introducing perturbed inputs during training. While this approach strengthens the model's defenses, it also comes with challenges, such as increased computational requirements, potential trade-offs between accuracy and robustness, and the risk of vulnerability to new or evolving attack strategies.

2.4.3 Random Noise

Random Noise as a Defense Mechanism is a simple technique presented by Cohen and colleagues [20] to defend machine learning models, particularly deep neural networks, against adversarial attacks. When the model is making predictions, the main aim is to introduce unpredictable, random fluctuations into the input data. The carefully planned modifications in adversarial cases may be disrupted by this, making the attack less likely to succeed.

How Random Noise Defense Works,

1. Adding Random Noise to Inputs

The input data (text, photos, etc.) is subjected to a small amount of random noise during the inference phase before being fed into the model. Typically, a probability distribution such as a uniform or Gaussian (normal) distribution—is used to sample the additional noise.

The noise is small enough to not significantly alter the original input, allowing the model to still make accurate predictions on clean (non-adversarial) inputs.

2. Disrupting Adversarial Perturbations

Adversarial examples are generated by introducing targeted changes to an input that take advantage of the model's vulnerabilities. By adding random noise, these intentional modifications are interfered with, reducing their ability to cause misclassification.

This can work because adversarial attacks often rely on precise changes to the input, and random noise introduces uncertainty that can "break" the effect of these changes.

3. Averaging Predictions over Multiple Noisy Versions

Making several inferences with various random noise samples added to the input and averaging the results is one method to increase the random noise defense's efficacy. By smoothing the decision border, this method, called randomized smoothing, might lessen the model's sensitivity to slight adversarial changes.

Why Random Noise Defense Can Be Effective,

1. Increases Model Robustness

Adding random, unpredictable changes to the input data makes it harder for adversarial samples to consistently trick the model. The attackers' little modifications are less likely to fool the noisy inputs in the same manner.

2. Simple and Easy to Implement

It is simple to implement and computationally cheap to add random noise to inputs. It doesn't call for intricate training processes or modifications to the model's architecture.

3. Randomized Smoothing as a Certified Defense

The predictions of the model are assured to be stable within a specific range of disturbances when random noise is used in conjunction with methods such as random smoothing. This is known as certified robustness.

Challenges and Limitations,

1. Balance Between Accuracy and Resilience

Excessive random noise may interfere with the original characteristics that are necessary for classification, which can lower the model's performance on undamaged data. Therefore, the amount of noise added, and the model's resilience need to be balanced.

2. Not a Fool proof Defense

While random noise can disrupt some adversarial attacks, it is not a complete defense. Adaptive adversaries can design attacks that account for the random noise, making the defense less effective.

3. Limited Protection Against Strong Attacks

For sophisticated adversarial attacks that involve larger perturbations or advanced techniques, random noise may not provide sufficient protection. Stronger defense mechanisms are needed for more robust adversarial defense.

Random Noise as a defense mechanism involves adding small, random noise to input data during inference to disrupt adversarial perturbations. It is a simple, computationally inexpensive approach that can increase model robustness by making it harder for adversarial examples to succeed. While effective for some attacks, it has limitations and may not fully protect against stronger or adaptive adversarial methods. Combining random noise with techniques like random smoothing can enhance the defense by providing certified robustness.

5.4.4 Neural Cleanse

Neural Cleanse, as described by Cohen, J. et al. [20], is a defense mechanism intended to identify and lessen trojan or backdoor attacks in machine learning models, especially deep neural networks. Malicious triggers are embedded into the model during training as part of backdoor attacks, which make the model respond improperly (misclassify inputs) when the trigger is present but function normally on clean inputs. The goal of Neural Cleanse is to locate and close these covert backdoors.

How Neural Cleanse Works,

1. Reverse Engineering Potential Triggers

Neural Cleanse operates by attempting to uncover possible backdoor triggers. It does so by identifying minimal changes (triggers) that, when applied to different inputs, consistently lead the model to categorize those inputs as belonging to a specific target class. For every class in the model, it seeks the smallest alteration that can turn various inputs into ones that the model classifies as the designated class.

2. Measuring Anomaly Scores

Once potential triggers are reverse engineered for all classes, Neural Cleanse calculates the size of each trigger. The intuition is that, for a backdoor attack, the trigger for the compromised class will be significantly smaller than triggers for other classes because the model was intentionally trained to be sensitive to that specific pattern.

An anomaly index is computed by comparing the size of each trigger to the median size of all the triggers. If the size of a particular trigger is much smaller than the others, it suggests a potential backdoor for that class.

3. Outlier Detection

To detect the presence of a backdoor, Neural Cleanse uses an outlier detection technique on the anomaly index values. If an anomaly score is significantly higher than a threshold, it indicates that a backdoor might exist for that particular class.

4. Mitigating the Backdoor (Fine-Pruning)

Once a backdoor is detected, Neural Cleanse can be used to mitigate the backdoor. One approach is fine pruning, which involves retraining the model to "unlearn" the backdoor trigger by removing the neurons or layers associated with the trigger while keeping the model's overall accuracy intact.

Another approach involves adding the triggers discovered to the data used for training the model. This allows the model to learn to resist their effects through retraining.

Why Neural Cleanse Works,

1. Explores All Classes for Backdoors

Systematically checking all classes for possible triggers, Neural Cleanse increases the chances of finding hidden backdoors regardless of which class has been compromised.

2. Leverages Anomaly Detection

The approach uses anomaly detection to identify unusually small perturbations that can trigger misclassifications, which is a characteristic feature of backdoor attacks.

Challenges and Limitations,

1. Sophisticated Backdoor Attacks

Neural Cleanse may be less effective against advanced backdoor attacks that use multiple or complex triggers, or if the backdoor is spread across multiple classes instead of targeting a single class.

2. High Computational Cost

The process of reverse-engineering possible triggers can be resource-intensive, particularly when dealing with large models that have numerous classes.

3. Possible False Positives

There is a risk of false positives, where benign behaviors might be flagged as potential backdoors. Setting an appropriate threshold for anomaly detection is crucial.

In addition, Chen et al. [17] describes Neural Cleanse as a defense mechanism that detects backdoor attacks by reverse-engineering potential triggers for each class and identifying unusually small triggers that could indicate a backdoor. It measures anomaly scores and uses outlier detection to identify classes with backdoors. Once detected, it mitigates the backdoor through techniques like fine pruning. While effective in many cases, it may struggle with sophisticated or multi-trigger backdoor attacks.

CHAPTER 3

PROPOSED METHOD

This study's primary goal is to put strategies in place and assess how well they guard against adversarial assaults, in which malevolent actors try to trick fraud detection technologies. The focus is on actually implementing these defenses and seeing how well they can protect fraud detection systems from misleading tactics designed to manipulate machine learning models. By applying these protective techniques to existing models, this research seeks to determine how effective they are in practical scenarios and to offer useful knowledge on how to make fraud detection systems stronger against adversarial threats.

when conducting research on credit card fraud detection in an adversarial environment, several steps can be outlined. Here's a suggested sequence of actions,

1. Data Collection

Gather a comprehensive collection of credit card transaction data that closely reflects actual real-world scenarios, encompassing both legitimate and fraudulent activities. This dataset needs to be varied, capturing the intricacies of different transaction behaviors and patterns.

2. Feature Selection

Use feature selection techniques to find and maintain the most important variables in the credit card dataset. This process is essential for enhancing the model's precision and computational effectiveness, particularly when dealing with feature-rich datasets. The specified features ought to be significant in properly differentiating between real and fraudulent transactions.

3. Simulation of Adversarial Environment

Create an adversarial setting to evaluate how well current credit card fraud detection models can withstand attacks. This involves deliberately introducing fake fraudulent transactions, altering existing ones, or applying other adversarial methods designed to mislead the models.

4. Model Evaluation

Place the existing credit card fraud detection algorithms in an adversarial environment. Analyze performance indicators like F1-score, recall, accuracy, and precision to determine how effective they are in these challenging circumstances. Analyze the models' ability to adjust to and protect themselves against hostile assaults.

5. Implementation of Defense Mechanism

Create and apply a range of defense strategies to improve the models' accuracy and resilient against adversarial threats. These strategies could involve integrating anomaly

detection methods, using ensemble techniques, or investigating new approaches specifically aimed at mitigating the effects of adversarial manipulations.

6. Evaluation of Defense Mechanisms

Assess the success of the implemented defense strategies by measuring improvements in model accuracy, sensitivity to fraudulent transactions, and overall performance. Compare these results with the baseline models operating under adversarial conditions. This method enables an in-depth evaluation of credit card fraud detection models under adversarial conditions, providing valuable understanding that aids in developing more robust and precise fraud detection solutions.

3.1 Data Collection and Feature Selection

This study's data set consists of September 2013 credit card transactions involving European cardholders. There are 284,807 transaction occurrences in this dataset that were recorded throughout a two-day period. Interestingly, 492 of these transactions were found to be fraudulent. Since fraudulent transactions only make up 0.172% of all transactions, the dataset is incredibly unbalanced. Generally speaking, no transaction infrastructure provider outsources customer information to outside parties. However, after feature selection as a worldwide requirement, VISA outsources old data sets for research purposes.

The study's dataset consists solely of numerical characteristics that have had their dimensionality reduced through the technique of Principal Component Analysis (PCA). Due to confidentiality limitations, specific information about the original features and further context is not available. The PCA transformation produces the modified variables, or V1 through V28, while leaving the 'Time' and 'Amount' attributes unchanged. While the 'Amount' feature shows the amount of each transaction after accounting for transaction cost, the 'Time' option shows the number of seconds that have elapsed since the dataset's first transaction. The 'Class' target variable tells you if a transaction is fraudulent (1) or legal (0). Figure 3.1 provided a layout sample for this dataset.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	
1	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15
2		0	-1.3598071	-0.0727812	2.53634674	1.37815522	-0.3383208	0.46238778	0.23959855	0.0986979	0.36378697	0.09079417	-0.5515995	-0.6178009	-0.9913898	-0.3111694	1.46
3		0	1.19185711	0.26615071	0.16648011	0.44815408	0.06001765	-0.0823608	-0.078803	0.08510165	-0.2554251	-0.1669744	1.61272666	1.06523531	0.48909502	-0.1437723	0.63
4		1	-1.3583541	-1.3401631	1.77320934	0.37977959	-0.5031981	1.80049938	0.79146096	0.24767579	-1.5146543	0.20764287	0.62450146	0.06608369	0.71729273	-0.1659459	2.34
5		1	-0.9662717	-0.185226	1.79299334	-0.8632913	-0.0103089	1.24720317	0.23760894	0.37743587	-1.3870241	-0.0549519	-0.2264873	0.17822823	0.50775687	-0.2879237	-0.6
6		2	-1.1582331	0.87773675	1.54871785	0.40303393	-0.4071934	0.09592146	0.59294075	-0.2705327	0.81773931	0.75307443	-0.8228429	0.53819555	1.34585159	-1.1196698	0.17
7		2	-0.4259659	0.96052304	1.14110934	-0.1682521	0.42098688	-0.0297276	0.47620095	0.26031433	-0.5686714	-0.3714072	1.34126198	0.35989384	-0.3580907	-0.1371337	0.51
8		4	1.22965763	0.14100351	0.04537077	1.20261274	0.19188099	0.27270812	-0.005159	0.08121294	0.46495999	-0.0992543	-1.4169072	-0.1538258	-0.7510627	0.16737196	0.05
9		7	-0.6442694	1.41796355	1.07438038	-0.492199	0.94893409	0.42811846	1.12063136	-3.8078642	0.61537473	1.24937618	-0.6194678	0.29147435	1.75796421	-1.3238652	0.6
10		7	-0.8942861	0.2861572	-0.1131922	-0.2715261	2.66959866	3.72181806	0.37014513	0.85108444	-0.3920476	-0.4104304	-0.7051166	-0.1104523	-0.2862536	0.07435536	-0.3
11		9	-0.3382618	1.11959338	1.04436655	-0.2221873	0.49936081	-0.2467611	0.65158321	0.06953859	-0.7367273	-0.3668456	1.01761447	0.83638957	1.00684351	-0.4435228	0.1
12		10	1.44904378	-1.1763388	0.91385983	-1.3756667	-1.9713832	-0.6291521	-1.4232356	0.04845589	-1.7204084	1.62665906	1.19964395	-0.6714398	-0.5139472	-0.095045	0.23
13		10	0.38497822	0.61610946	-0.8742997	-0.0940186	2.92458438	3.31702717	0.47045467	0.53824723	-0.5588946	0.30975539	-0.2591156	-0.3261432	-0.0900467	0.36283237	0.92
14		10	1.24999874	-1.2216368	0.38393015	-1.2348987	-1.4854195	-0.7532302	-0.689405	-0.2274872	-2.0940106	1.32372927	0.22766623	-0.242682	1.20541681	-0.3176305	0.72
15		11	1.06937359	0.28772213	0.82861273	2.71252043	-0.178398	0.33754373	-0.0967169	0.11598174	-0.2210826	0.46023044	-0.7736569	0.32338725	-0.0110759	-0.1784852	-0.6
16		12	-2.7918548	-0.3277708	1.64175016	1.76747274	-0.1365884	0.80759647	-0.4229114	-1.9071075	0.75571291	1.15108699	0.84455547	0.79294395	0.37044809	-0.7349751	0.40
17		12	-0.752417	0.34548542	2.05732291	-1.4686433	-1.1583937	-0.0778498	-0.6085814	0.00360348	-0.436167	0.74773083	-0.7939806	-0.7704067	1.047627	-1.0666037	1.10
18		12	1.10321544	-0.0402962	1.26733209	1.28909147	-0.7359972	0.28806916	-0.5860568	0.18937971	0.78233289	-0.2679751	-0.4503113	0.93670771	0.70838041	-0.4686473	0.35
19		13	-0.4369051	0.91896621	0.92459077	-0.7272191	0.91567872	-0.1278674	0.70764161	0.08796236	-0.6652714	-0.7379798	0.32409781	0.27719211	0.25262426	-0.2918965	-0.1
20		14	-5.4012577	-5.4501478	1.18630463	1.7362388	3.04910588	-1.7634056	-1.5597377	0.16084175	1.23308974	0.34517283	0.91722987	0.97011672	-0.2665678	-0.4791299	-0.5
21		15	1.49293598	-1.0293457	0.45479473	-1.4380259	-1.5554341	-0.7209611	-1.0806641	-0.0531271	-1.9786816	1.63807604	1.07754241	-0.6320465	-0.4169572	0.05201052	-0.0
22		16	0.69488478	-1.3618191	1.02922104	0.8341593	-1.1912088	1.30910882	-0.8785859	0.44529013	-0.4461958	0.56852074	1.01915061	1.2983287	0.42048027	-0.372651	-0.8
23		17	0.96249607	0.32846103	-0.1714791	2.10920407	1.12956557	1.69603769	0.10771161	0.52150216	-1.1913111	0.72439631	1.69032992	0.40677358	-0.9364213	0.98373942	0.71
24		18	1.16661638	0.50212009	-0.0673003	2.26156924	0.42880419	0.08947352	0.24114658	0.13808171	-0.9891624	0.92217497	0.74478579	-0.5313773	-2.1053465	1.1268701	0.00
25		18	0.24749113	0.27766563	1.18547084	-0.0926025	-1.314394	-0.150116	-0.946365	-1.6179351	1.5440714	-0.8298806	-0.5831995	0.52493323	-0.4533753	0.08139309	1.5
26		22	-1.9465251	-0.0449005	-0.4055701	-1.0130573	2.9419677	2.9550534	-0.0630631	0.85554631	0.0499669	0.57374251	-0.0812565	-0.215745	0.04416063	0.03389776	1.19

Figure 3.1

Sample of collected feature selected data.

3.2 Training and Evaluating Fraud Detection Models

To demonstrate how the existing fraud detection system operates, I apply multiple algorithms including Random Forest, Logistic Regression, Support Vector Machine (SVM), and Decision Tree. These methods are essential for analyzing and classifying credit card transactions, aiding in the identification of potentially fraudulent activities. Employing a range of algorithms allows for a comprehensive evaluation of the system's performance, providing a balanced perspective on its ability to detect fraud effectively.

3.3 Simulation of Adversarial Environment

This study uses a strategy that applies several attack techniques, including DeepFool, Elastic Net, and zeroth-order optimization, to mimic adversarial circumstances. These methods purposefully change input data in order to deceive machine learning models. The goal is to assess how stable and resilient the models are to different kinds of hostile perturbations. By including these attacks, the study seeks to pinpoint the models' shortcomings and investigate how they respond to hostile pressure. This approach provides crucial information about the system's possible weaknesses in the event of intentional data manipulation, enabling a thorough evaluation of the models' resistance to such dangers.

3.4 Implementation of Defense Mechanism

This study uses a comprehensive assessment framework that incorporates many well-established methods to increase the robustness of machine learning models. Neural Cleanse, Random Noise Injection, General Adversarial Training, and Defensive Distillation are the defensive mechanisms employed in this work.

3.5 High Level Architecture

In the proposed solution in Figure 3.2, the presentation layer consists of three wizards: the data upload wizard, the normal environment setting wizard, the adversarial environment setting wizard, and the defense mechanism wizard. The business logic layer includes a framework library with four modules: the data load module, feature selection module, training in a normal environment module, and training in an adversarial environment module. The data layer contains temporary data storage.

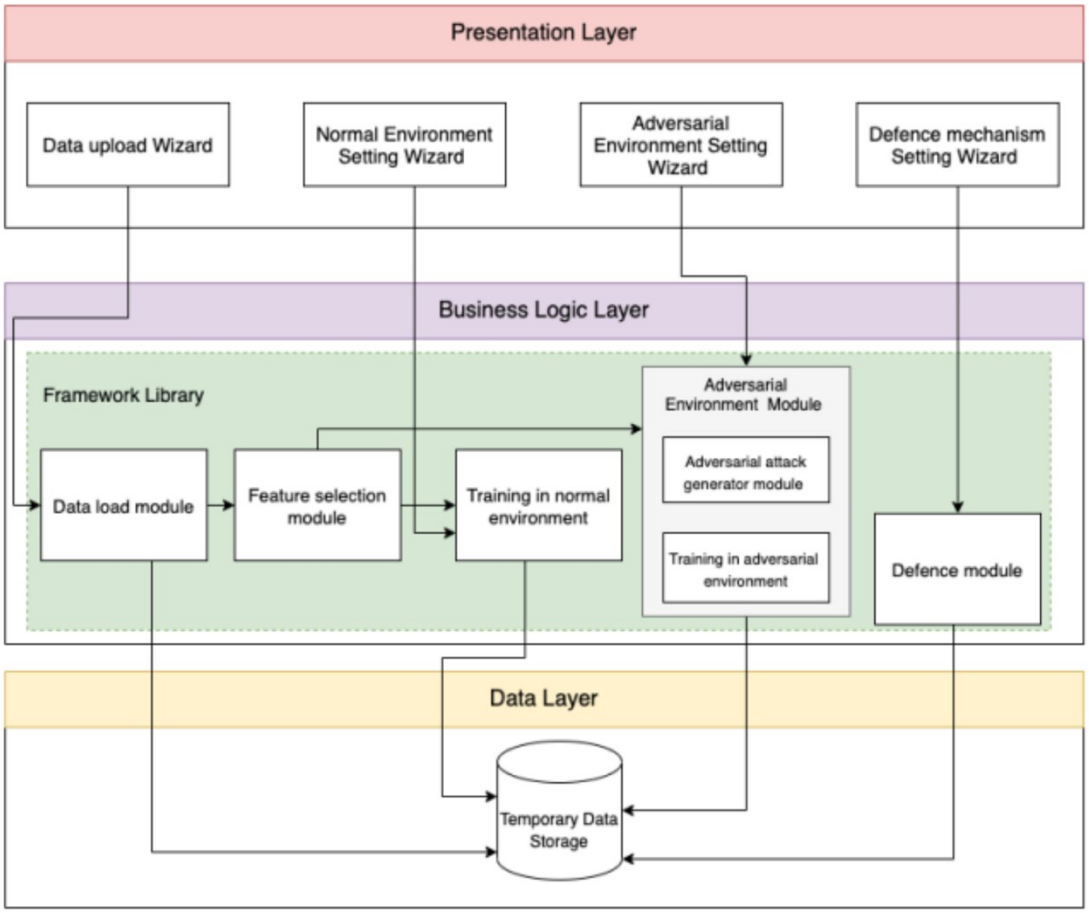

Figure 3.2

High level architecture of proposed solution

The multi-layered architecture of a system intended to facilitate the development of machine learning models, especially in hostile contexts, is depicted in Diagram 3.2. The Presentation Layer, Business Logic Layer, and Data Layer are the three main layers that make up the system.

The Presentation Laye provides the user interface through a set of wizards that guide the user in performing various tasks. These include a Data Upload Wizard for uploading datasets, a Normal Environment Setting Wizard for configuring standard model training, an Adversarial Environment Setting Wizard for setting up training under adversarial conditions, and a Defense Mechanism Setting Wizard for configuring defensive strategies against adversarial attacks.

The Business Logic Layer contains the core functional components and is subdivided into different modules. It starts with the Framework Library, which includes a Data Load Module for ingesting data, a Feature Selection Module for selecting relevant input features, and a Training in Normal Environment module for standard model training. Additionally, it includes an Adversarial Environment Module composed of an Adversarial Attack Generator Module, which creates adversarial data, and a Training in Adversarial Environment module that uses this data for robust model training. A Defense Module is also present to implement and evaluate strategies

against adversarial threats. These modules are interconnected, allowing data to flow from ingestion through processing to training and defense.

The Data Layer is responsible for storing intermediate and temporary data used by the system. It facilitates communication between the various modules by providing temporary data storage that supports the processes carried out in the business logic layer.

Overall, this architecture supports a comprehensive workflow for data ingestion, feature selection, model training in both normal and adversarial environments, and the application of defense mechanisms, making it well-suited for research and development in adversarial machine learning.

The user representation presented in the figure 3.2 above outlines the process. Initially, users upload a data file that has undergone feature selection and then select their desired machine learning algorithm from options like Random Forest, Logistic Regression, Support Vector Machine (SVM), or Decision Tree to begin training the model.

The model's performance is evaluated in an adversarial environment once it has been trained. Users may simulate adversarial scenarios and assess the resilience of the model by utilizing techniques like Deep Fool, Elastic Net, and zeroth-order optimization.

Following adversarial attacks, users can implement defense mechanisms and assess accuracy changes. This is done by applying techniques such as Defensive Distillation, General Adversarial Training, Random Noise, and Neural Cleanse.

3.6 Summary

This research investigates and puts into practice four different ways to defend against adversarial attacks: Defensive Distillation, General Adversarial Training, Random Noise, and Neural Cleanse. Each of these methods was chosen because it takes a unique approach to making models stronger against adversarial threats. Defensive Distillation uses adjusted probability scores during training to make the model's decision boundaries smoother, which reduces how sensitive it is to adversarial changes. General Adversarial Training enhances model robustness by incorporating adversarial samples during training, enabling the model to accurately classify both legitimate and manipulated inputs. Random Noise Injection introduces minor, random perturbations to the input data during prediction, which interferes with the specific patterns crafted in adversarial attacks, thereby reducing their effectiveness. Finally, Neural Cleanse focuses on finding and reducing the impact of backdoor attacks by figuring out potential triggers and identifying unusual behaviors. These four methods were selected to create a well-rounded defense strategy that addresses different kinds of adversarial attacks, including those where the attacker knows the model's details, those where they don't, and backdoor attacks. My implementation aims to evaluate

how well these defenses work on their own and when used together, to find the best ways to secure machine learning models.

CHAPTER 4

IMPLEMENTATION

The Flask framework was used to build the system, which is divided into five primary phases: feature selection, data upload, model training, adversarial environment simulation, and adversarial defense. In the Data Upload Stage, users can easily upload a credit card dataset via a user-friendly web interface. After the data is uploaded, the Feature Selection Stage processes and normalizes the data before selecting relevant features for model training. During the Model Training Stage, users can train various machine learning models, including Linear Regression, Random Forest, SVM, and Decision Trees, with the flexibility to choose the desired model.

Once the model is trained, the Adversarial Environment Simulation Stage introduces adversarial attacks like Zeroth-Order Optimization, DeepFool, and Elastic Net to simulate potential real-world attacks, enabling assessment of the model's effectiveness prior to and following the attacks. Finally, in the Adversarial Defense Stage, defense techniques such as Defensive Distillation, General Adversarial Training, Random Noise, and Neural Cleanse are applied to strengthen the model's resilience. The system then presents the updated accuracy scores, offering a comprehensive platform for training, testing, and defending models against adversarial threats.

Figure 4.1

Activity flow of the implementation

4.1 Dataset

This study uses 284,807 credit card transactions from European cardholders during a 48-hour period in September 2013. Only 492 of these (0.172%) were discovered to be fraudulent, resulting in a very imbalanced sample. This significant class imbalance makes it challenging to evaluate model performance correctly as typical accuracy metrics may not adequately reflect true effectiveness.

The basic numerical properties of the transactions were changed using Principal Component Analysis (PCA). Important patterns extracted from the original data were recorded by each of the 28 new features produced by this transformation and assigned the names V1 through V28. Two characteristics, however, were unaffected by PCA: "Time" and "Amount." The 'Time' feature tracks the number of seconds since the dataset's initial transaction's 'Amount' feature shows the transaction's monetary value. Principal Component Analysis (PCA), the transactions' initial numerical attributes were transformed. Each of the 28 new features that were created by this transformation and given the labels V1 through V28 captured important patterns that were taken from the original data. However, PCA did not alter two features: "Time" and "Amount." The 'Time' feature tracks the number of seconds since the dataset's initial transaction, while the 'Amount' feature shows the transaction' monetary value.

In this research, the original numerical properties of the transactions using Principal Component Analysis (PCA). With the designations V1 through V28, each of the 28 new features produced by this modification was able to identify significant patterns extracted from the original data. There were two features that PCA did not change, though: "Time" and "Amount." 'Time' records the number of seconds since the dataset's initial transaction, while 'Amount' shows the transaction's monetary value.

4.2 Build a fraud detection models

Using sklearn's train_test_split function, the dataset is split in this study so that 80% of the oversampled training data (X_train_over and y_train_over) is used to train the models and 20% is reserved for testing to see how well they generalize. A fixed random seed of 42 was used to ensure that the results could be reproduced. Throughout the experiment's several runs, this setup keeps the data samples' random distribution constant. This consistency yields similar results and lessens differences in model assessment that can result from chance.

While xDataTest and yTest include the features and labels for testing, xDataTrain and yTrain contain the features and labels for training. Four pieces of data are produced by this process. This type of data separation is essential because it keeps the models from simply memorising the training data (overfitting) and allows to examine how well they perform on freThis study examined a number of machine learning methods, such as Decision Trees, Random Forest, Support Vector Machines, and Linear Regression, to ascertain which are best at detecting fraud. Comparing many models helps determine the best accurate and dependable method for detecting fraudulent transactions in a dataset when legitimate instances greatly outnumber fraudulent ones. The models

were evaluated using the Area Under the Precision-Recall Curve (AUPRC), which is especially useful for datasets that have a notable imbalance between valid and fraudulent transactions.sh, unknown data.

4.2.1 Random Forest

Using the RandomForestClassifier from the scikit-learn library, the Random Forest classifier was used in this study as one of the techniques for identifying fraud. The model was evaluated on a different set, X_test and y_test, which it had not come across during training, after being trained on the balanced dataset, X_train_over, with labels y_train_over. The classifier was initialised and fitted to the training data as part of the training procedure. Following training, predictions were performed on test data, and the model's performance was assessed using a number of important metrics, including the Matthews Correlation Coefficient (MCC), accuracy, precision, recall, and F1-score. Precision and recall explicitly evaluate the model's capacity to detect fraudulent transactions, whereas accuracy gauges the general correctness of predictions. For unbalanced datasets like this one, where fraud instances are far less common than real ones, MCC provides a trustworthy assessment, while the F1-score provides a trade-off between accuracy and recall. Together, these metrics provide a comprehensive assessment of the model's ability to identify fraudulent activity, highlighting the need of considering factors other than accuracy when evaluating the model's ability to handle the tiny proportion of fraudulent transactions.

4.2.2 Logistic Regression

The second machine learning method used in this study to detect fraudulent transactions is logistic regression. The balanced dataset X_train_over and its corresponding labels, y_train_over, are used to train the model. The model's ability to detect fraud is then assessed using the distinct test sets X_test and y_test. The training and evaluation procedure, which includes fitting the model to the training data and using it to forecast results on the test data, is A number of measures, including as accuracy, precision, recall, F1-score, and Matthews Correlation Coefficient, are computed in order to fully evaluate the model's performance. The difficulty presented by the dataset's significant imbalance and dearth of fraud instances was addressed by the selection of these markers. Two important plots are created for visual assessment: the Confusion Matrix, which makes use of Seaborn's heatmap to clearly display the counts of true positives, true negatives, false positives, and false negatives; and the Receiver Operating Characteristic (ROC) curve, which shows the trade-off between true positive rate and false positive rate across different thresholds. Together, these assessment tools offer a thorough grasp of how well Logistic Regression identifies fraud in the dataset, demonstrating its suitability for this use ascribed in the LoRegression class.

4.2.3 Support Vector Machine (SVM)

Support Vector Machine (SVM) was the third classification technique used in this investigation. The best boundary (hyperplane) to divide data points into the appropriate classes is sought for by SVM, a supervised learning method. To facilitate the probability computations required for ROC curve drawing, the implementation made use of the SVC class from the sklearn.svm package, with the parameter probability=True enabled. The model was trained using the training subset of the dataset (xTrain, yTrain) following The model's performance was assessed using important metrics including accuracy, precision, recall, F1-score, and Matthews Correlation Coefficient (MCC) after predictions were made on the test set after training. The model's ability to detect fraudulent transactions while reducing false positives and false negatives was evaluated using these metrics. To give a comprehensive view of the categorization results and help identify areas where the model works well or needs to be improved, a confusion matrix was also used. This thorough analysis highlights the SVM technique's potential for usage in real-world applications by demonstrating how effectively it handles the class imbalance issues that are common in fraud detection. The information was divided.

4.2.4 Decision Tree

In this study, the Decision Tree classifier which was developed using the sklearn.tree module's DecisionTreeClassifier was used to identify fraudulent transactions. In order to make classification judgements, decision trees supervised learning models use feature values to create a hierarchical tree structure and divide the dataset into smaller groups. Using the training data, the classifier learnt the patterns that differentiate authentic transactions from fraudulent ones. Following training, predictions were generated on the test data (xTest), and a number of important measures, such as accuracy, precision, recall, F1-score, and Matthews Correlation Coefficient (MCC), were used to assess the model's performance.

These measures offer a thorough understanding of how well the system detects fraud while lowering missed detections and false alarms. The ability of the model to distinguish between fraud and non-fraud instances was further evaluated by plotting a Receiver Operating Characteristic (ROC) curve, which shows the trade-off between true positive and false positive rates. To better clarify the categorization results, a confusion matrix was also created to display the proportion of accurate and inaccurate responses for each class. In the end, this comprehensive analysis shows that the Decision Tree model is well calibrated and appropriate for real-world fraud detection applications.

4.3 Simulation of Adversarial Environment

In order to assess the classifiers' resilience to attacks, the study now simulates a hostile environment. This section employs three distinct attack techniques: Elastic Net, DeepFool, and Zeroth-Order Optimization (ZOO). DeepFool seeks to apply minimal changes to the input to shift the model's classification boundary, resulting in incorrect

classifications. Zeroth-Order Optimization, on the other hand, avoids using gradient calculations and creates deceptive samples by strategically altering input data according to the predictions.

In the end least, the Elastic Net assault blends two regularisation strategies (L1 and L2), utilising their respective advantages in producing sparse modifications and general efficacy to evade detection. In order to determine how these assaults impact the classifiers' performance and provide crucial information about their resilience to adversarial inputs, this research phase simulates these attacks.

4.3.1 Zeroth-Order Optimization (ZOO)

At this stage of the research, the model is tested by simulating an adversarial environment using the Zoo Adversarial Attack. Using the Adversarial Robustness Toolbox (ART), this black-box attack creates misleading instances without requiring an understanding of the model's core operations. By trying to generate inaccurate predictions, the Zoo Attack assesses how well the model performs when faced with challenging, altered inputs. This process mimics real-world adversarial assaults, when inputs are deliberately designed to deceive the model. To assess the model's performance, the Zoo Adversarial Attack technique creates fictitious samples from the training and testing datasets. Metrics like accuracy, recall, and F1-score are then used to assess the model's performance with these modified inputs, along with visual aids like confusion matrices and ROC curves. This approach offers a thorough evaluation of the model's response to altered data, highlighting both its advantages and disadvantages in difficult situations. These simulations are essential for comprehending how the model responds to detrimental inputs, spotting errors, and evaluating the efficacy of defense tactics meant to increase the model's resilience and dependability in actual situations..

4.3.2 Deep Fool

A classification model's resilience to adversarial situations is assessed using the DeepFool attack. This method causes the model to misclassify incoming data by making minuscule, nearly imperceptible changes to it. Its main goal is to determine the smallest modification required to change the model's forecast. Because of this, it's a useful tool for determining how responsive the model is to slight changes in its input characteristics. Because the attack is used on both training and testing datasets, it is possible to thoroughly examine the model's stability and how adversarial inputs affect its accuracy.

Through the creation of changed instances from both training and test data, this simulation assesses the model's performance under hostile influence. To evaluate the model's performance under these difficult circumstances, important assessment metrics are employed, such as precision, recall, F1-score, and the ROC curve. Any decrease in accuracy is highlighted by comparing the outcomes of the original model with the one subjected to adversarial inputs. Confusion matrices and ROC curves are two examples of visualization tools that show how the model responds to various data

types during an assault. In order to strengthen the model's security and reliability against actual attacks, this phase of the research use the DeepFool technique to pinpoint the model's weakest areas and suggests enhancements like adversarial training or defensive measures..

4.3.3 Elastic Net attack

The ElasticNet attack is used in this section of the research to evaluate the classification model's resilience to adversarial challenges. This technique modifies the input data in small, deliberate ways, which may cause the model to anticipate things incorrectly. ElasticNet effectively generates such misleading inputs by balancing sparsity and smoothness through the combination of L1 and L2 regularization. Metrics including accuracy, recall, F1-score, and ROC curves are used to assess the model's performance following the assault on both the training and testing data. To ascertain the extent to which the accuracy of the original model is impacted, these assessments are contrasted with its output. The confusion matrix provides a clearer understanding of the classification errors that are being targeted, while the ROC curve aids in visualising the trade-off between true positives and false positives. This evaluation shows how the model responds to hostile inputs and points out possible problems with it. The results provide recommendations for improving the model, such employing adversarial training to fortify its resistance against comparable attacks in practical situations.

4.4 Implement of Defense Mechanism Technique

Defensive distillation, general adversarial training, random noise injection, and neural cleanse are the four types of defence mechanisms that were employed in this study to increase the robustness of my model against adversarial attacks. Since defensive distillation uses soft target labels for training, this method seeks to lower the model's accuracy by minor adjustments.

4.4.1 Defensive Distillation

Defensive distillation is used in this study to improve the algorithms' capacity to identify credit card fraud with greater accuracy. Initially, transactional data is used to train a deep neural network (DNN) to detect both authentic and fraudulent transactions. Rather than generating rigid classifications, the model generates "soft labels," which are probability distributions that represent the machine's level of confidence in every forecast. More detailed information is provided by these soft labels than by straightforward class labels. Following that, a second model is trained with these soft labels, enabling it to gain knowledge from the intricate insights that the previous model was able to extract. By improving the second model's generalization abilities and assisting in the identification of subtle patterns, this technique leads to more accurate and reliable fraud detection..

These probability-based soft labels are then used instead of the initial binary fraud indicators to train a Random Forest classifier. The intention is for this more straightforward model to gain knowledge from the deep neural network's more illuminating outputs. By this distillation process, the simpler model uses a lot less computing power while simulating the sophisticated network's decision-making behaviour. Thus, by adjusting to the soft labels, the student model is designed to replicate the performance of the complicated model, allowing it to process information more quickly and use less resources while achieving a similar level of prediction accuracy.

In the last step, a test data set is used to evaluate the original and distilled models' capacity to detect fraudulent credit card transactions. Their performance is assessed using important classification metrics including accuracy, precision, recall, and F1-score. In addition to being more computationally efficient, it is expected that the distilled (simpler) model would perform similarly to the more complicated model in terms of prediction accuracy. These findings demonstrate the benefits of defensive distillation and demonstrate how it may improve fraud detection abilities while simultaneously increasing the model's usefulness for real-time application by fusing strong performance with lower processing requirements.

4.4.2 General Adversarial Training

This study employs General Adversarial Training as a defensive method to enhance a model's ability to detect fraudulent credit card transactions. In order to identify fraud, the initial stage in the process is to prepare the dataset, which involves normalizing transaction values and deleting characteristics like "Time" and "Amount" after suitable scaling. The data is then separated into input characteristics and labels, with the 'Class' attribute denoting the transaction's legitimacy. StandardScaler is then used to standardize features in order to optimize performance once the dataset has been divided into training and testing sets..

A starting point This data is used to train the Random Forest classifier, and accuracy, precision, recall, and F1-score are used to gauge how effective it is. Following the establishment of this baseline, adversarial sample inputs that are slightly altered to trick the model are produced. Both original and adversarial data are used to retrain the model, improving its ability to identify and categorize both authentic and modified transactions. The performance of the revised model is then compared with the original using the same evaluation criteria and a confusion matrix to display prediction accuracy across all categories. This technique significantly boosts the model's robustness, enabling it to handle intricate attempts to evade fraud detection in real-world scenarios.

4.4.3 Random Noise

Adding random fluctuations, such Gaussian noise, to the test data is another method of evaluating a machine learning model's capacity to manage unforeseen interruptions in credit card fraud detection. Following standard preparation procedures, such as feature scaling to normalize transaction volumes, the dataset is divided into training and testing sets. The prepared data is used to train a Random Forest classifier, and its initial performance is evaluated using key performance indicators such as accuracy, precision, recall, and F1-score to determine how well it distinguishes between real and fraudulent transactions in real-world scenarios. Then, to mimic real-world inconsistencies like data input errors or transmission problems, Gaussian noise is purposefully added to the test data at a certain level. This additional noise is used to evaluate the model's ability to continue making accurate decisions in spite of flaws. Following this modification, the model is assessed once more using the same performance criteria, and the classification outcomes are shown using a confusion matrix. In real-world fraud detection tasks, where data may not always be clean or consistent, the model's reliability and robustness to random disturbances are demonstrated by comparing these results with the original, noise-free results.

4.4.4 Neural Cleanse

The use of Neural Cleanse as a defensive strategy to recognize and lessen the impact of adversarial attacks in credit card fraud detection systems is examined in this study. Finding inputs that have been cleverly changed to deceive machine learning algorithms is how Neural Cleanse operates. This method takes a dataset of credit card transactions, normalizes the 'Amount' feature, then divides the data into training and testing sets in order to construct a neural network. The output layer of the network, which has three hidden layers, employs a sigmoid function to classify transactions as either legitimate or fraudulent. The defensive system imitates opponent strategies by making unexpected modifications to the test data. The model identifies suspicious circumstances with significant prediction variations as possible hazards by comparing predictions from the original and updated data. A confusion matrix is used to visualize the classification findings, and evaluation metrics including accuracy, precision, recall, and F1-score are used to gauge how effective this approach is. This technique improves the model's resistance and accuracy in identifying fraud, even in hostile environments, by fortifying its capacity to identify manipulated inputs. In the end, Neural Cleanse helps create safer and more dependable fraud detection systems..

4.6 Summary

The usefulness of many machine learning models, including LR, RF, SVM, and Decision Trees, in detecting fraudulent credit card transactions is investigated in this paper. These models are trained on a dataset of transaction records and assessed using common assessment metrics like as accuracy, precision, recall, and F1-score. Several

adversarial attack techniques, including as DeepFool, Elastic Net, and Zeroth-Order Optimization (ZOO), are used to the models after the first assessment. These techniques try to fool the models by subtly altering the input data. The robustness of the models is then evaluated by looking at how these adversarial perturbations affect the models' performance indicators..

A number of defense techniques, including as Defensive Distillation, General Adversarial Training, Random Noise Injection, and Neural Cleanse, are used to combat these hostile threats. These methods are intended to make the models stronger and the attacks less effective. The models are retested to determine whether their performance has improved after implementing these defenses. The results of this study will clarify how various defense methods improve the resilience of fraud detection algorithms and provide insightful information about thwarting adversarial attacks in practical applications.

The five steps of the procedure described above data upload, feature selection, model training, adversarial environment simulation, and adversarial defence implementation are depicted in Figure 4.2.

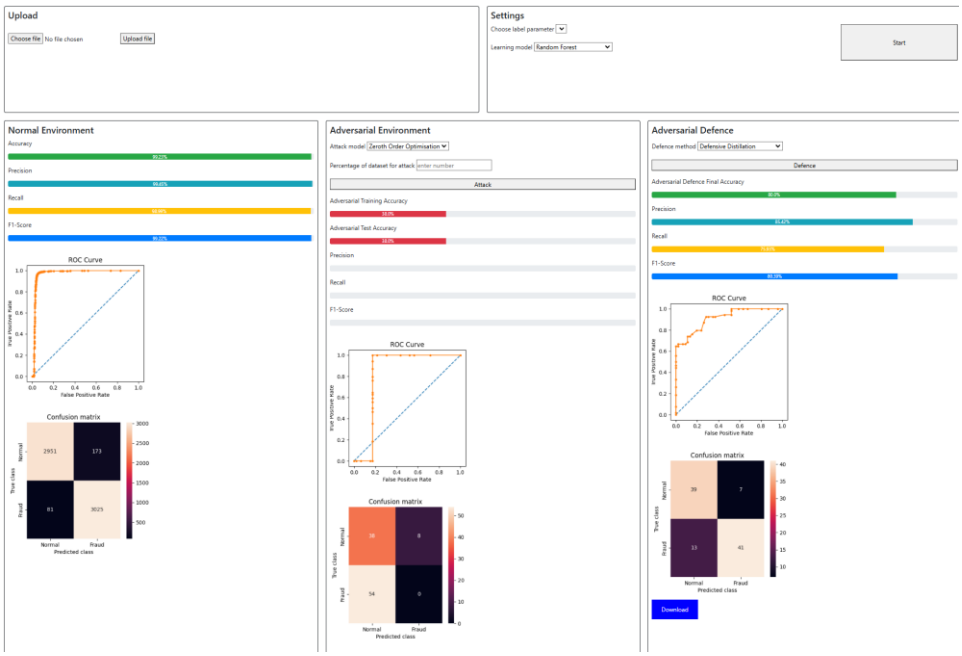

Figure 4.2

Screenshot of final implementation

CHAPTER 5

EVALUATION

This research presents the final evaluation of black-box, white-box, and hybrid environments, along with the results of implementing appropriate defense mechanisms.

5.1 Evaluation Metrics

This study assesses the performance of machine learning models by looking at a variety of performance indicators, providing a comprehensive picture of their advantages and disadvantages. Every chosen indicator draws attention to a distinct facet of the model's behavior and offers a different perspective on its overall efficacy. A fuller picture of the model's capacity to identify trends and provide precise forecasts is provided by the combined examination of these indicators.

1. Training Accuracy: This measure shows how well the model was able to categorize the training data. It displays the percentage of accurate forecasts in the training set. Although it is sometimes viewed as a desirable thing, great training accuracy does not ensure successful performance on fresh, untested data. Overfitting is a situation when the model has grown too specific to the training data.
2. Test Accuracy: Performance on a separate dataset that wasn't used to train the test set is known as test accuracy. It provides an indication of the model's generalizability to new, untested data. Achieving a balance between high training and test accuracy is crucial to preventing the model from overfitting the training data while retaining a high degree of predictive ability.
3. Precision: Precision measures how well a model detects positive cases out of all the cases it properly predicted to be positive. It is computed by taking the entire number of positive predictions (including false positives and true positives) and dividing it by the number of true positive findings. This statistic is particularly important when misclassifying negatives as positives might have expensive or bad consequences.
4. Recall (Sensitivity): Memory is a measure of a model's ability to identify every single positive event in a dataset. By dividing the number of properly predicted positive cases by the total number of correctly detected positives plus the positives that the model missed, it may be computed. This statistic is particularly crucial when ignoring actual positive cases might have detrimental or dangerous consequences.
5. F1-Score: The F1-Score is a metric that computes the harmonic mean of accuracy and recall integrating them into a single metric. This score takes into account both false positives and false negatives to offer a fair evaluation of a model's

performance. It provides a clearer perspective of the model's overall performance and is especially helpful when the data is spread unevenly.

6. ROC Curve (Receiver Operating Characteristic Curve): This illustrates the relationship between the true positive rate, also referred to as sensitivity, and the false positive rate, which is calculated by deducting specificity from one another, across a range of decision thresholds. The AUC provides a summary score that reflects the model's overall ability to differentiate between classes at all threshold values. Better categorization performance is indicated by a bigger AUC.

Together, these evaluation metrics offer a comprehensive picture of the model's performance, including its accuracy, dependability, and ability to recognize real favorable circumstances. Examining accuracy, recall, and the F1-Score from different angles may help one gain a more complete understanding of the model's advantages and disadvantages. Furthermore, the ROC curve provides important information about the model's ability to distinguish between classes using a variety of classification criteria..

5.2 Evaluation of the Implementation

In here I evaluated the Blackbox, white box and hybrid environments.

5.2.1 Blackbox Environment

Within a black-box setting, this study utilizes the zeroth-order optimization (ZOO) method and assesses its effectiveness on previous implemented models. Following this, protective strategies including Defensive Distillation, General Adversarial Training, Random Noise, and Neural Cleanse are implemented to counteract the ZOO attack, and the outcomes are systematically evaluated.

5.2.1.1 Random Forest

This section describes RF algorithm is subjected to a zeroth-order optimization (ZOO) attack, followed by the application of defensive mechanisms such as Defensive Distillation, General Adversarial Training, Random Noise, and Neural Cleanse.

At the outset, the Random Forest model is tested under standard conditions, where it attains an impressive accuracy of 99.33%, highlighting its capability to correctly classify the vast majority of cases. With a precision of 99.58%, the model demonstrates a strong ability to reduce false positive errors, ensuring that nearly all positive predictions are accurate. Its recall rate of 99.07% shows that it successfully identifies almost all true positive instances, missing very few. The F1-score of 99.33% indicates a well-balanced performance between precision and recall, confirming the model’s effectiveness in making accurate predictions and reliably differentiating between positive and negative samples. The related confusion matrix and ROC curve for this evaluation are displayed in Figure 5.1.

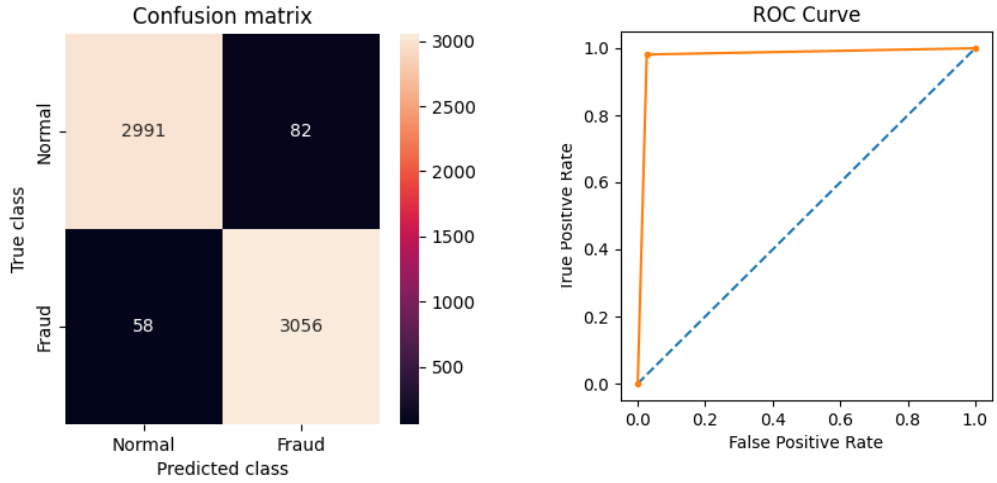

Figure 5.1

Confusion matrix and ROC with RF

When the ZOO attack is applied on 1% of the dataset, the model's performance noticeably declines. Testing and training accuracy drop to 52.0% and 41.0%, respectively. Precision falls to 21.43%, suggesting an increase in false positives, while recall falls to 7.5%, reflecting the model's reduced ability to detect real positive instances. The F1-score falls to 11.11%, indicating that the model is less able to balance accuracy and recall. Figure 5.2 displays the confusion matrix and ROC curve at this level.

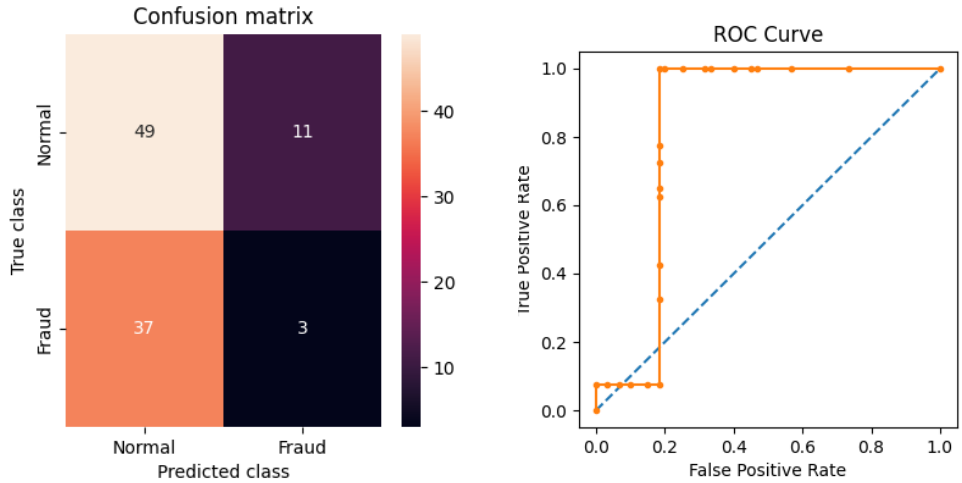

Figure 5.2

After the ZOO attack to RF model Confusion matrix and ROC curve

1. Application for Defensive Distillation

After employing defensive distillation, the F1-score increased to 81.82%, recall to 90.0%, accuracy to 84.0%, and precision to 75.0%. At his stage, the confusion matrix and ROC curve are shown in Figure 5.3.

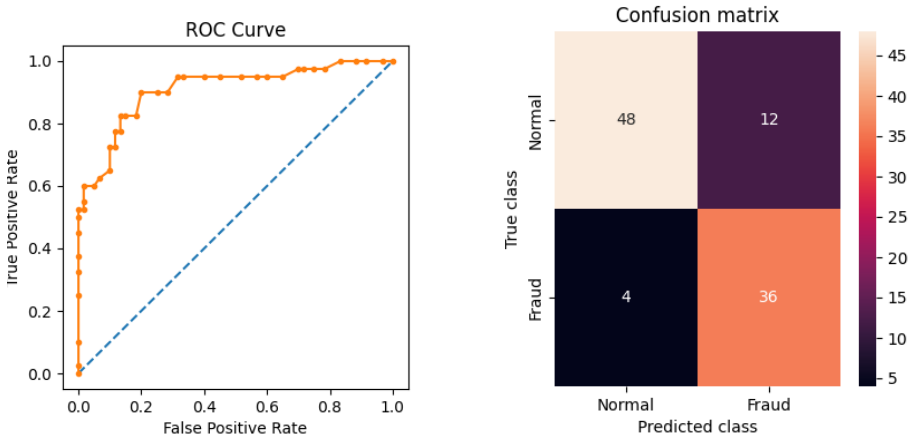

Figure 5.3

After Applied Defensive Distillation Confusion matrix and ROC curve

2. Application for General Adversarial Training

Accuracy rose to 73.0%, precision to 79.49%, recall to 62.0%, and the F1-score to 69.66% following the implementation of General Adversarial Training. Figure 5.4 displays the confusion matrix and ROC curve at this point.

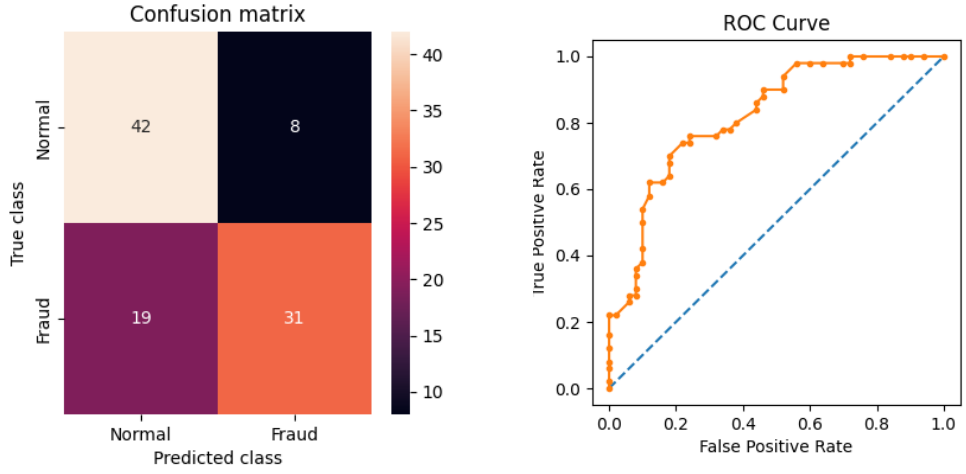

Figure 5.4

Confusion matrix and ROC curve after the ZOO attack to RF model with General Adversarial Training

3. Application of Random Noise

Accuracy rose to 74.0%, precision to 76.09%, recall to 70.0%, and the F1-score to 72.92% when Random Noise was applied. Figure 5.5 displays the confusion matrix and ROC curve at this point.

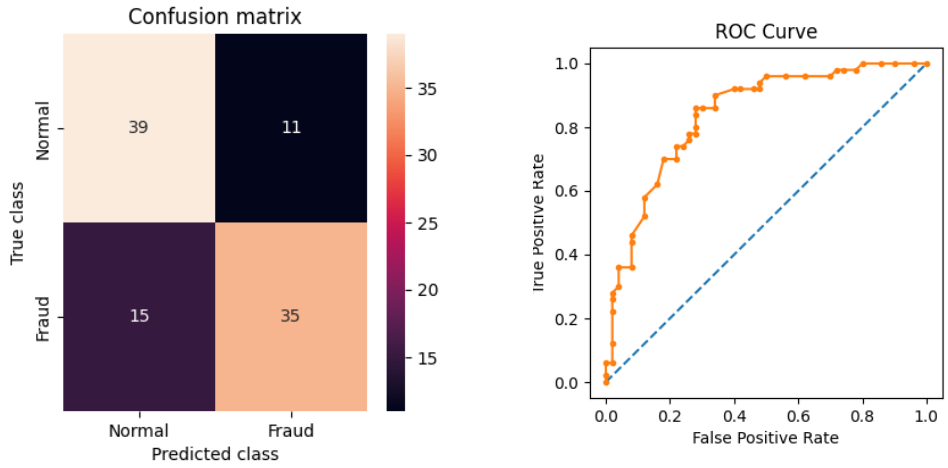

Figure 5.5

Confusion matrix and ROC curve after the ZOO attack to RF model with Random Noise

4. Application of Neural Cleanse

Accuracy rose to 75.0%, precision to 80.49%, recall to 66.0%, and the F1-score to 72.53% following the application of Neural Cleanse. Figure 5.6 displays the confusion matrix and ROC curve at this point.

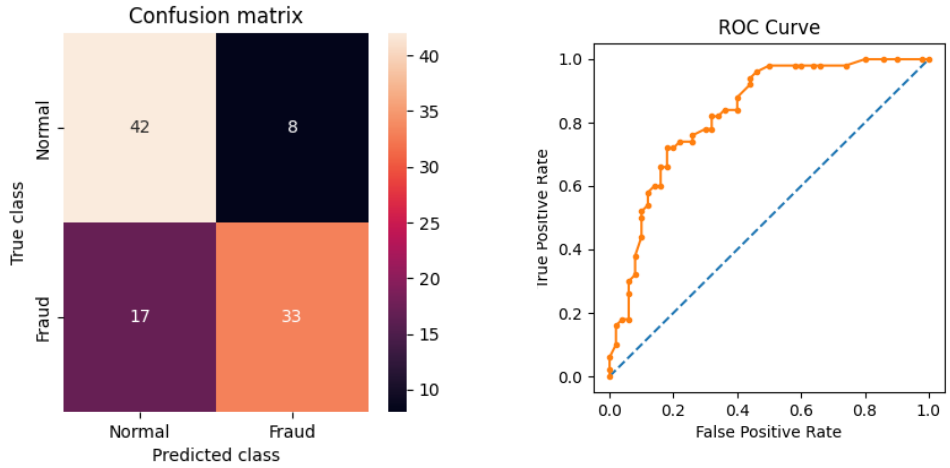

Figure 5.6

Confusion matrix and ROC curve after the ZOO attack to RF model with Neural Cleanse

5.2.1.2 Logistic Regression

From here, the Logistic Regression algorithm is subjected to a zeroth-order optimization (ZOO) attack, followed by the application of defensive mechanisms such as Defensive Distillation, General Adversarial Training, Random Noise, and Neural Cleanse.

The logistic regression model performed well under typical circumstances, achieving 94.86% accuracy, 93.71% precision, 96.18% recall, and 94.93% F1-score. The model's strong performance in accurately recognising both positive and negative cases is demonstrated by these figures. Figure 5.7 displays the associated confusion matrix and ROC curve.

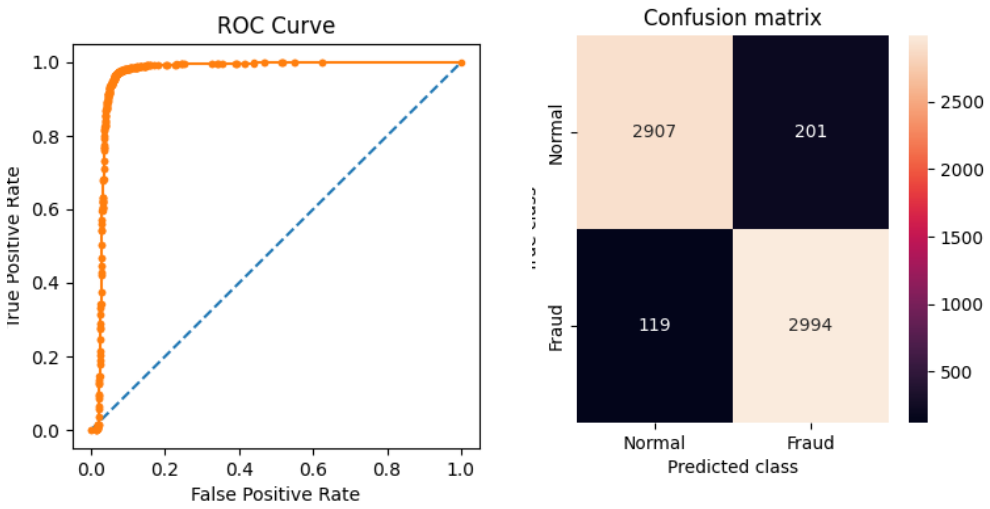

Figure 5.7

Confusion matrix and ROC with LR

When 1% of the data were subjected to the ZOO assault, the model's performance dramatically declined in comparison to its performance in a typical setting. The test accuracy decreased from 94.86% to 7.0%, while the training accuracy fell precipitously from 94.86% to 4.0%. The F1-score fell from 94.93% to 4.12%, recall fell from 96.18% to 3.33%, and precision fell from 93.71% to 5.41%. The confusion matrix and ROC curve for this stage are displayed in figure 5.8.

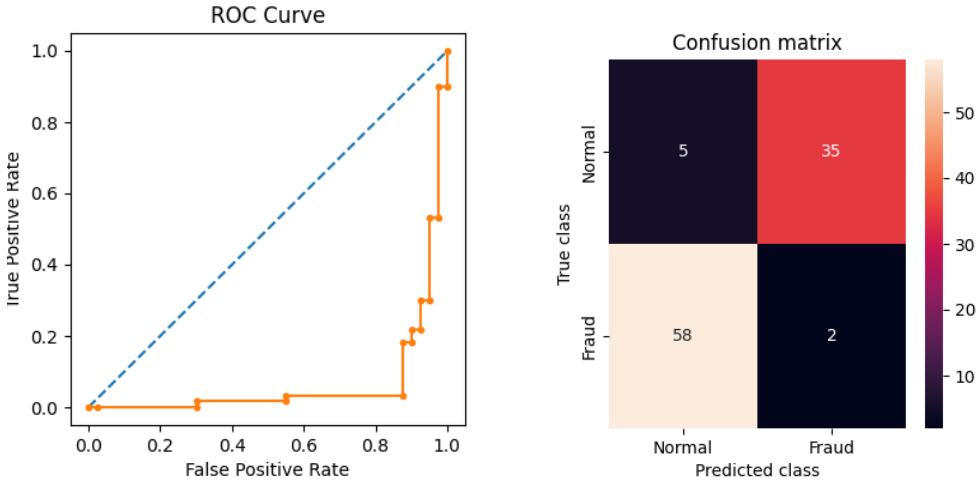

Figure 5.8

Confusion matrix and ROC curve after the ZOO attack to LR model

1. Application of Defensive Distillation

The model's performance significantly improved once the Defensive Distillation defence strategy was applied to it. The F1-score improved to 70.4%, recall improved to 73.33%, precision improved to 67.69%, and final accuracy increased to 63.0%. The confusion matrix and ROC curve at this point are displayed in Figure 5.9.

.

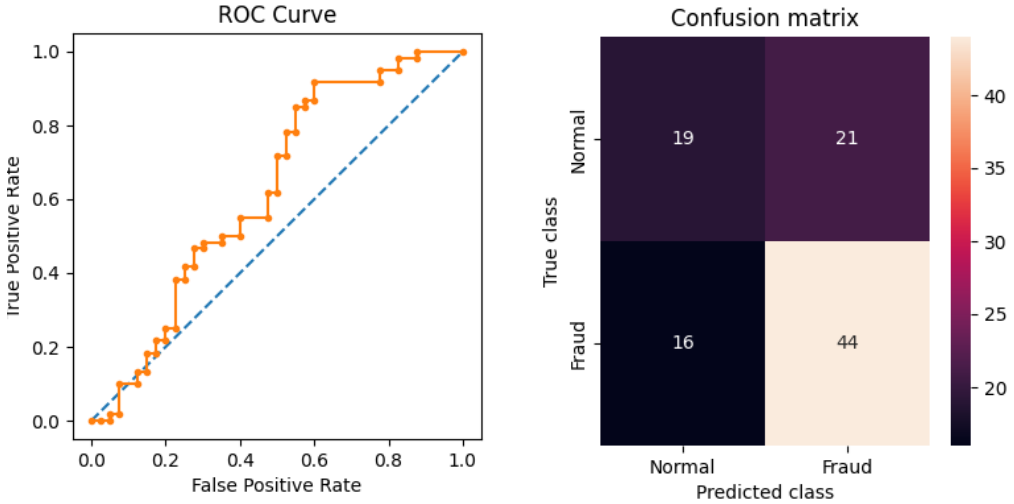

Figure 5.9

Confusion matrix and ROC curve of LR model after the Applied Defensive Distillation

2. Application of General Adversarial Training

The model obtained an accuracy of 57.0%, precision of 61.19%, recall of 70.69%, and F1-score of 65.6% after submitting an application for General Adversarial Training.

These findings show a modest increase in striking a balance between accurate predictions and real positives. The associated confusion matrix and ROC curve for this phase are shown in Figure 5.10.

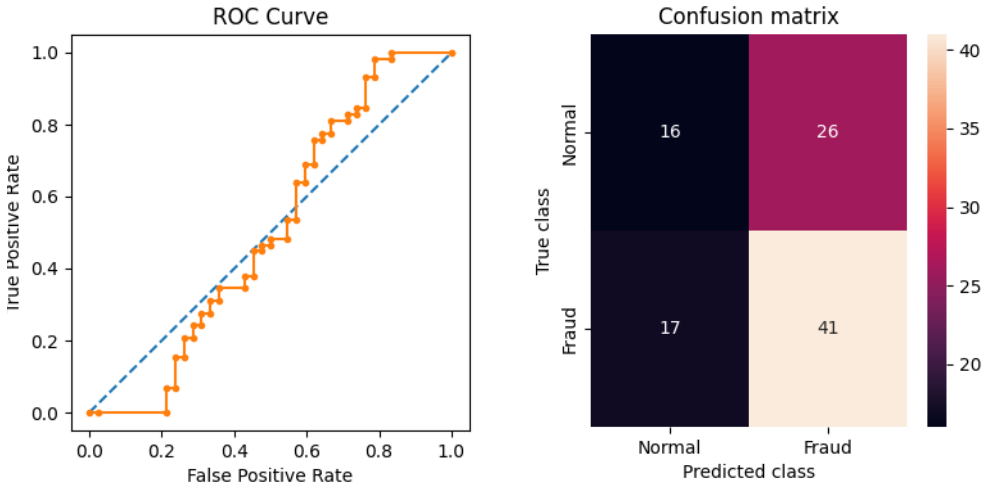

Figure 5.10

Confusion matrix and ROC curve of LR model after the Applied General Adversarial Training

3. Application of Random Noise

After adding random noise to the data, the model's performance declined, resulting in an F1-score of 52.63%, accuracy of 55.0%, precision of 53.19%, and recall rate of 52.08%. These signs show that the model's forecast reliability has declined. Figure 5.11 displays the ROC curve and confusion matrix for this level.

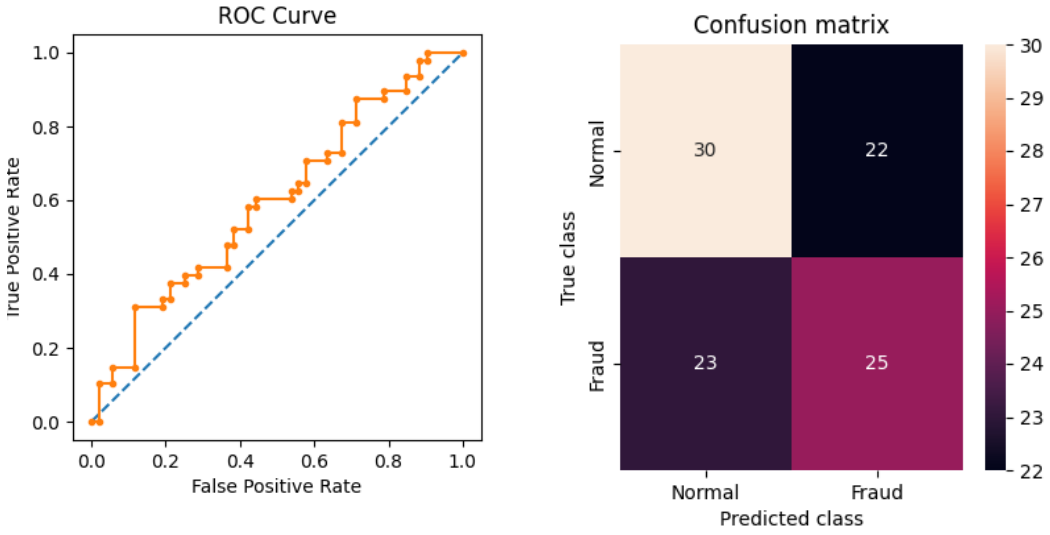

Figure 5.11

Confusion matrix and ROC curve of LR model after the Applied Random Noise

4. Application of Neural Cleanse

After applying the Neural Cleanse technique, the model's final accuracy, precision, recall, and F1-score were 53.0%, 53.45%, and 60.78%, respectively. These figures indicate a little increase in the model's ability to identify actual positive cases while maintaining the overall prediction equilibrium. Figure 5.12 displays the ROC curve and related confusion matrix for this level.

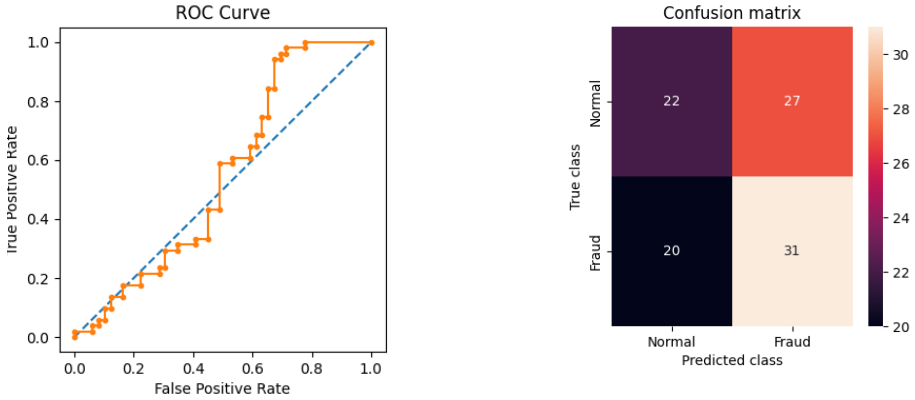

Figure 5.12

Confusion matrix and ROC curve of LR model after the Applied Neural Cleanse

5.2.1.3 Support Vector Machine (SVM)

This section details the process of applying a zeroth-order optimization (ZOO) attack to the SVM algorithm and subsequently implementing defense strategies such as Defensive Distillation, General Adversarial Training, Random Noise, and Neural Cleanse.

Under normal operating conditions, the SVM model produced an accuracy of 52.17%, a precision rate of 52.42%, a recall of 42.78%, and an F1-score of 47.11%. These metrics indicate the initial performance baseline of the model. Figure 5.13 displays the confusion matrix and ROC curve at this level.

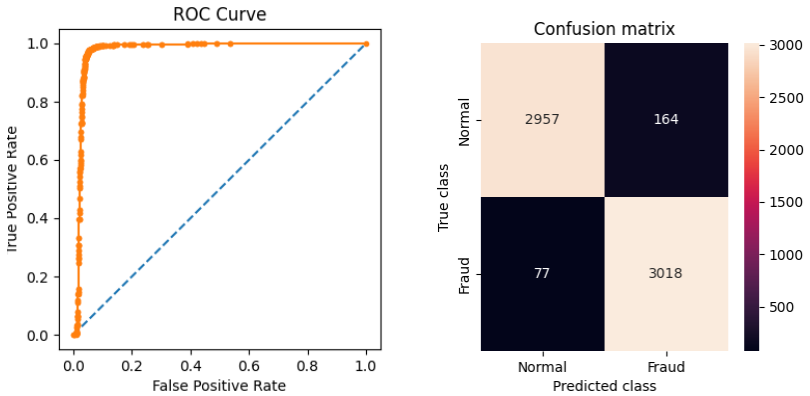

Figure 5.13

Confusion matrix and ROC with SVM

The SVM model's performance dramatically declined after 1% of the dataset was subjected to the ZOO attack. Both test and training accuracy decreased to 47.0% and 46.0%, respectively. The F1-score dropped to 41.76%, recall dropped to 35.85%, and precision dropped to 50.0%. The confusion matrix and ROC curve that show this decline in performance are shown in Figure 5.14.

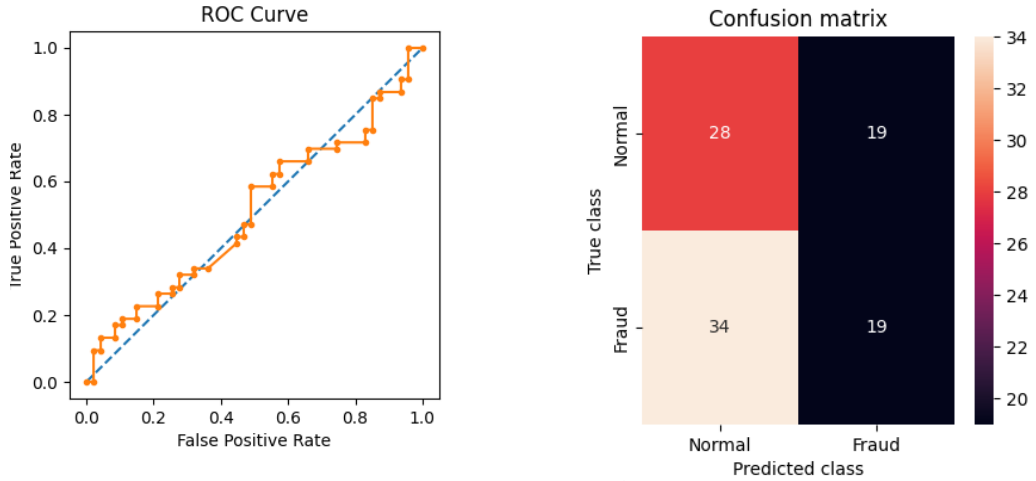

Figure 5.14

Confusion matrix and ROC curve after the ZOO attack to SVM model

1. Application of Defensive Distillation

The SVM model showed a small improvement in performance after using Defensive Distillation, with an F1-score of 43.37%, accuracy of 53.0%, precision of 60.0%, and recall of 33.96%. The ROC curve and confusion matrix for this phase are shown in Figure 5.15.

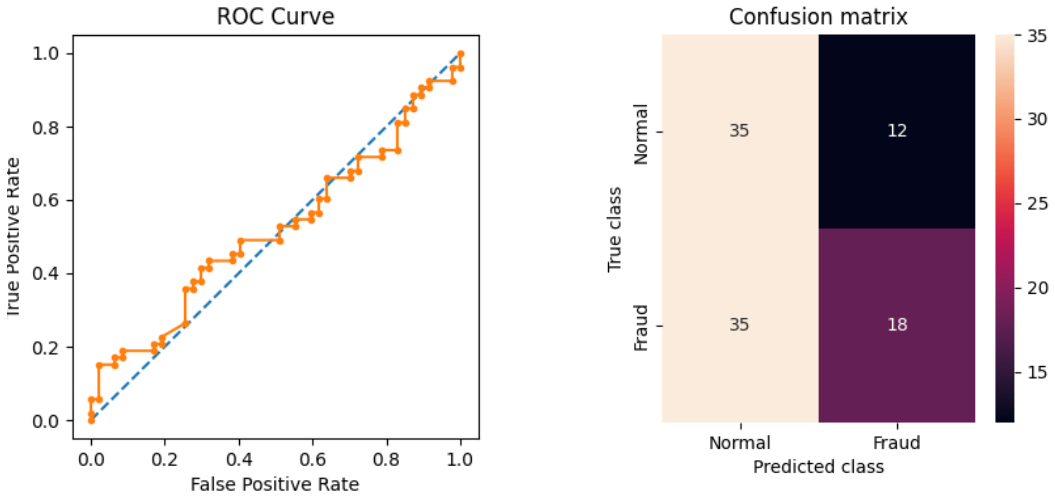

Figure 5.15

Confusion matrix and ROC curve of SVM model after the Applied Defensive Distillation

2. Application for General Adversarial Training

After using General Adversarial Training, the SVM model achieved a final accuracy of 54.0%, with precision, recall, and F1-score all at 39.47%. These findings are depicted in Figure 5.16's confusion matrix and ROC curve for this phase.

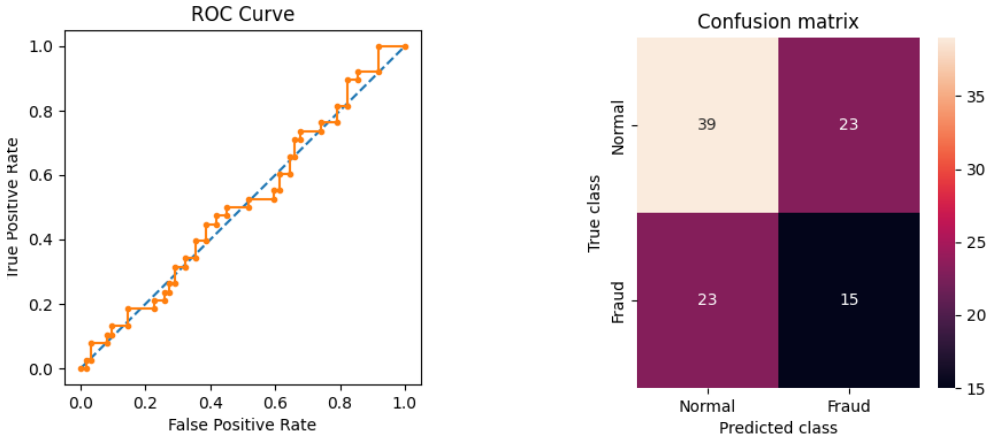

Figure 5.16

Confusion matrix and ROC curve of SVM model after the Applied General Adversarial Training

3. Application of Random Noise

With accuracy of 43.0%, precision of 52.38%, recall of 37.29%, and an F1-score of 43.56%, the SVM model's performance became erratic when Random Noise was included. The confusion matrix and ROC curve show these findings in Figure 5.17.

.

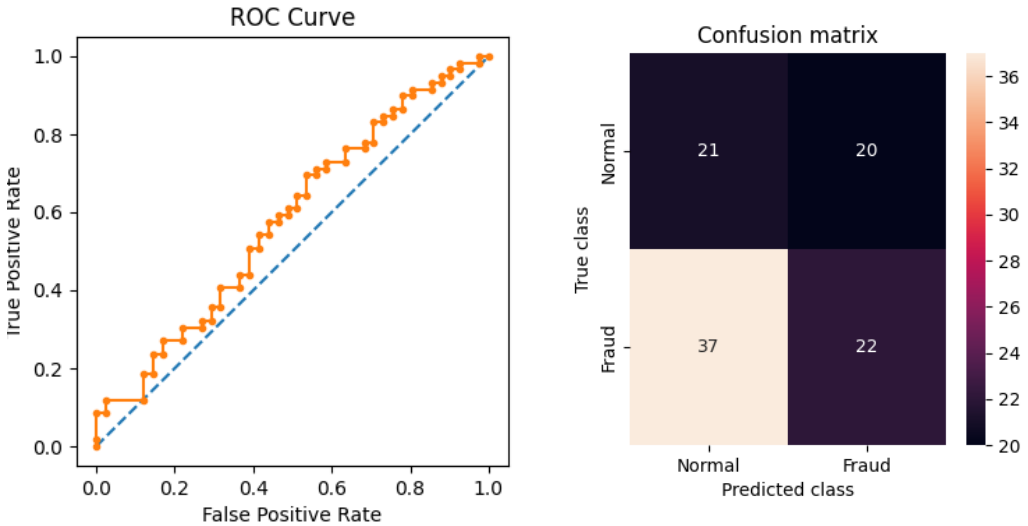

Figure 5.17

Confusion matrix and ROC curve of SVM model after the Applied Random Noise

4. Application of Neural Cleanse

The SVM model achieved a final accuracy of 50.0% when Neural Cleanse was applied, along with precision of 49.15%, recall of 59.18%, and F1-score of 53.7%. A visual representation of these results is given by the confusion matrix and ROC curve at this stage, which are shown in Figure 5.18.

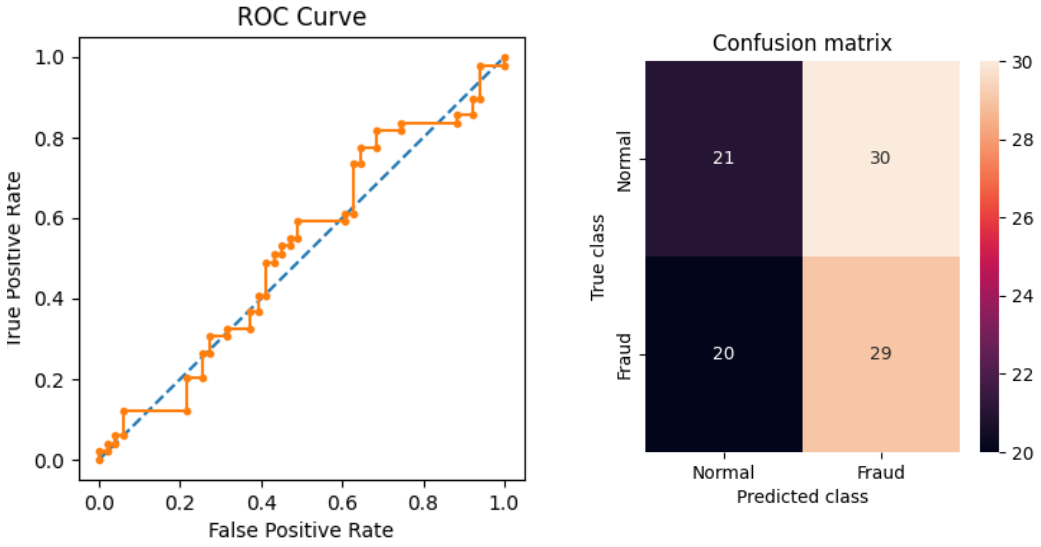

Figure 5.18

Confusion matrix and ROC curve of SVM model after the Applied Neural Cleanse

5.2.1.4 Decision Tree

This section assesses the Decision Tree algorithm's performance when subjected to a zeroth-order optimization (ZOO) attack and then examines its response after applying various defense strategies such as Defensive Distillation, General Adversarial Training, Random Noise, and Neural Cleanse, with relevant result.

Under normal circumstances, the Decision Tree (DT) model achieved 97.43% accuracy, 96.83% precision, 97.96% recall, and 97.40% F1-score. The paper's figure 5.19 shows the confusion matrix and ROC curve that represent this result.

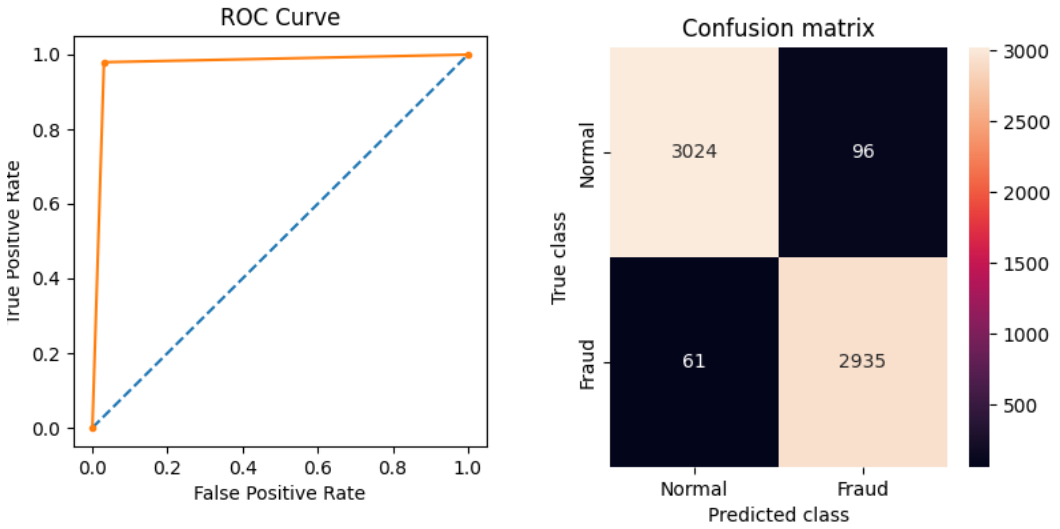

Figure 5.19

Confusion matrix and ROC with DT

Following a ZOO attack affecting 1% of the dataset, the Decision Tree model experienced a decline in performance. Training accuracy reduced to 74.0%, while test accuracy dropped to 76.0%. Precision reached 75.0%, recall decreased to 71.74%, and the F1-score fell to 73.33%. The confusion matrix and ROC curve illustrating these results are figure 5.20 in this section.

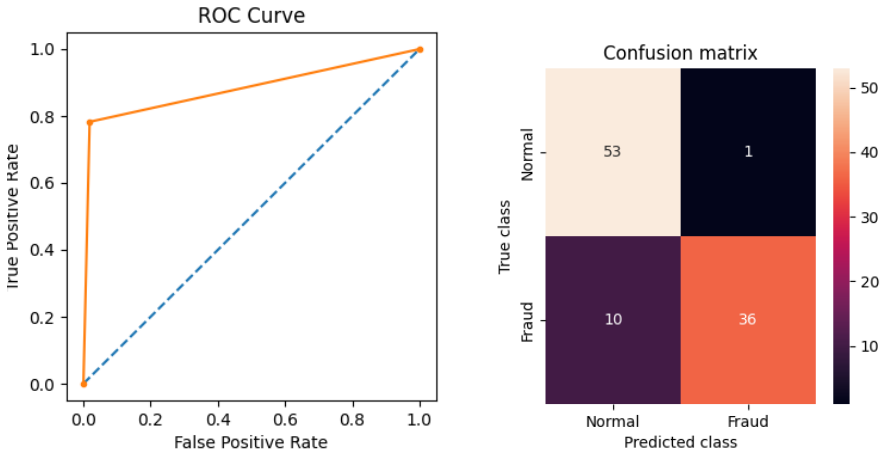

Figure 5.20

Confusion matrix and ROC curve after the ZOO attack to DT model

1. Application for Defensive Distillation

Following the implementation of Defensive Distillation, the F1-score grew to 76.29%, recall increased to 80.43%, accuracy increased to 77.0%, and precision reached 72.55%. The confusion matrix and ROC curve that show these findings are shown in figure 5.20.

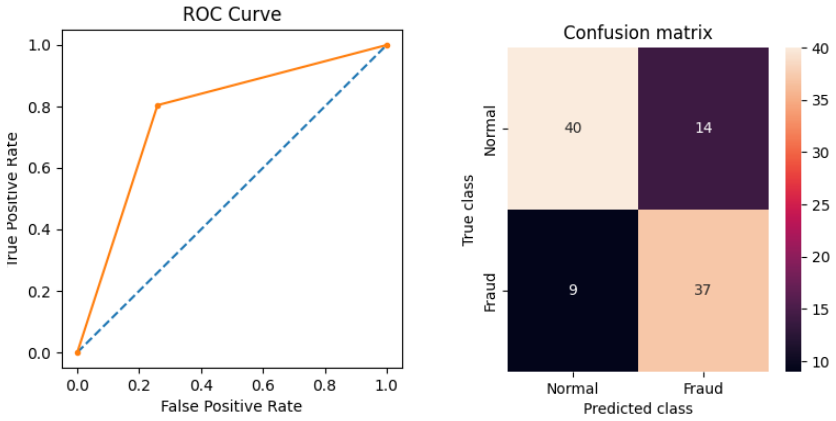

Figure 5.21

Confusion matrix and ROC curve of DT model after the Applied Defensive Distillation

2. Application for General Adversarial Training

General Adversarial Training improved accuracy to 85.0%, precision to 87.8%, recall to 78.26%, and F1-score to 82.76%. At this point, the ROC curve and confusion matrix are displayed in figure 5.22.

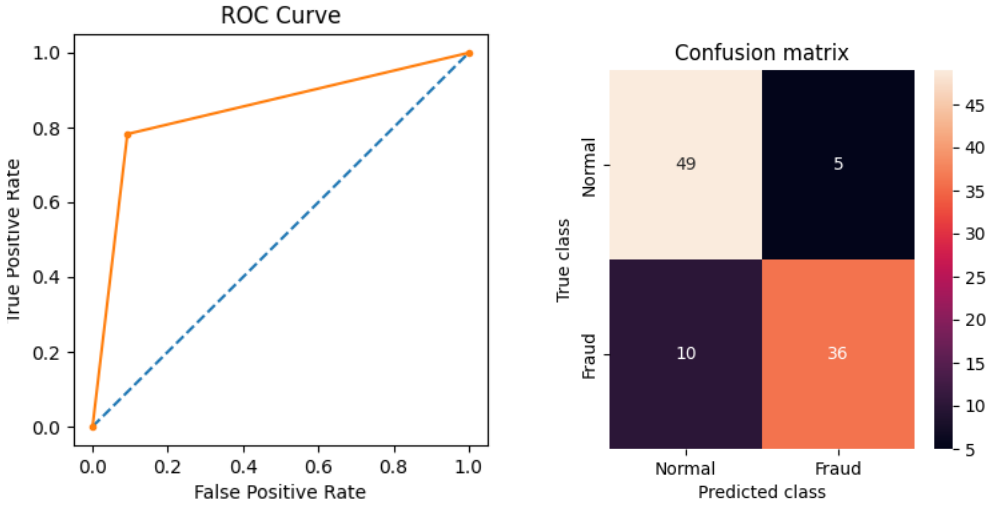

Figure 5.22

Confusion matrix and ROC curve of DT model after the Applied General Adversarial Training

3. Application of Random Noise

Accuracy changed to 81.0% Precision 81.4% Recall 76.09% F1-Score 78.65% with the use of Random Noise. At this point, the ROC curve and confusion matrix are displayed in figure 5.23.

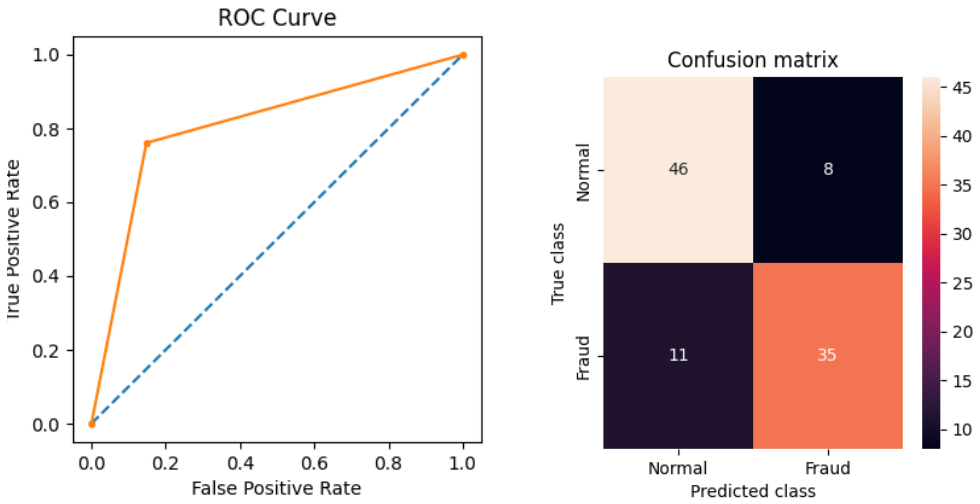

Figure 5.23

Confusion matrix and ROC curve of DT model after the Applied Random Noise

4. Application of Neural Cleanse

Accuracy improved to 83.0%, Precision 83.72%, Recall 78.26%, and F1-Score 80.9% following the use of Neural Cleanse. At this point, the confusion matrix and ROC are displayed in figure 5.24.

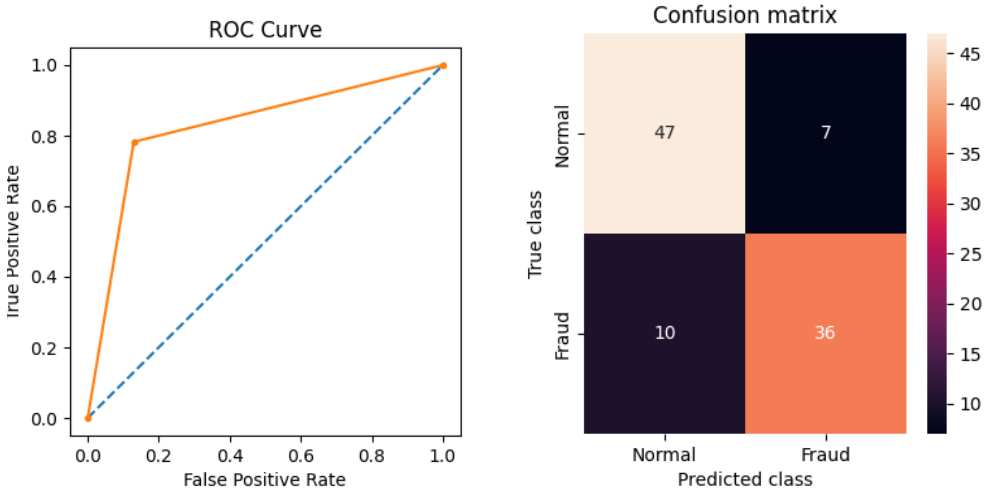

Figure 5.24

Confusion matrix and ROC curve of DT model after the Applied Neural Cleans

5.2.1.5 Summary view in Blackbox environment

Table 5.1

Comparative analysis results of black box environment

Algorithm	Measurement	Normal Environment	Adversarial Environment (ZOO)	Adversarial Defense			
				Defensive Distillation	General Adversarial Training	Random Noise	Neural Cleanse
RF	Accuracy	99.33	52.0	84.0	73.0	74.0	75.0
	Precision	99.58	21.43	75.0	79.49	76.09	80.49
	Recall	99.07	7.5	90.0	62.0	70.0	66.0
	F1-Score	99.33	11.11	81.82	69.66	72.92	72.53
LR	Accuracy	94.86	7.0	63.0	57.0	55.0	53.0
	Precision	93.71	5.41	67.69	61.19	53.19	53.45
	Recall	96.18	3.33	73.33	70.69	52.08	60.78
	F1-Score	94.93	4.12	70.4	65.6	52.63	56.88
SVM	Accuracy	52.17	46.0	53.0	54.0	43.0	50.0
	Precision	52.42	47.0	60.0	39.47	52.38	49.15
	Recall	42.78	50.0	33.96	38.34	37.29	59.18
	F1-Score	47.11	35.85	43.37	37.44	43.56	53.7
DT	Accuracy	97.43	74.0	77.0	85.0	81.0	83.0
	Precision	96.83	75.0	72.55	87.8	81.4	83.72
	Recall	97.96	71.74	80.43	78.26	76.09	78.26
	F1-Score	97.40	73.33	76.29	82.76	78.65	80.9

Four machine learning models RF, LR, SVM, and Decision Tree are compared in Table 5.1 after being tested under normal circumstances, during a ZOO adversarial attack, and with defenses in place. With an accuracy and F1-score of 99.33%, RF performs better than the others under typical conditions. With an accuracy of 97.43%, Decision Tree comes in second, while Logistic Regression scores a strong 94.86%. In contrast, the SVM model performed noticeably worse, achieving an accuracy of only 52.17%.

Under the ZOO attack, the performance of all algorithms declines sharply, with RF accuracy dropping to 52.0% and LR plummeting to 7.0%. SVM experiences a slight decline compared to its already low baseline, while DT demonstrates relative resilience, maintaining a higher accuracy of 74.0% and F1-score of 73.33%, showcasing its robustness against adversarial perturbations. These findings emphasize how susceptible conventional models are to adversarial attacks.

When defense mechanisms are applied, performance improves across the board. Defensive Distillation and Neural Cleanse are particularly effective for RF and DT, with DT achieving a robust accuracy of 83.0% under Neural Cleanse. LR and SVM also see modest improvements, though their metrics remain below those of RF and DT. General Adversarial Training provides strong gains for DT, which reaches 85.0% accuracy. Overall, RF and DT consistently outperform other models, demonstrating their robustness and adaptability when paired with appropriate defenses.

5.2.2 Whitebox Environment

In a white-box environment, this research experiments with the DeepFool attack applied RF, LR, SVM, and Decision Tree models. Following this, various defense techniques are tested, including Defensive Distillation, General Adversarial Training, Random Noise Injection, and Neural Cleanse.

According to table 5.2 with evaluation of different machine learning models under normal and adversarial environments, particularly with a focus on the DeepFool attack in a white-box setting. RF and DT models are highlighted for their inherent robustness against DeepFool attacks due to their non-differentiable nature and discontinuous decision boundaries, which render the attack inapplicable. Both models exhibit high performance in a normal environment, with RF achieving 99.33% accuracy and DT closely following at 97.43%, showcasing their reliability in standard scenarios.

Logistic Regression (LR) and Support Vector Machines (SVM) demonstrate notable vulnerabilities under adversarial attacks. For LR, the DeepFool attack significantly reduces accuracy from 94.86% in a normal environment to 6.0%. However, adversarial defenses, such as General Adversarial Training and Neural Cleanse, improve LR's accuracy to 91.0% and 93.1%, respectively, highlighting the effectiveness of these strategies in mitigating adversarial impact. SVM, in contrast, shows relatively low performance across environments, achieving 52.17% accuracy in a normal setting and experiencing marginal changes under adversarial attacks and defenses, indicating limited resilience and adaptability.

Table 5.2

Comparative analysis results of white box environment

Algorithm	Measurement	Normal Environment	Adversarial Environment (DeepFool)	Adversarial Defense			
				Defensive Distillation	General Adversarial Training	Random Noise	Neural Cleanse
RF	Accuracy	99.33	Can not execute Deep Fool attack due to their **non-differentiable nature** and **discontinuous decision boundaries**				
	Precision	99.58					
	Recall	99.07					
	F1-Score	99.33					
LR	Accuracy	94.86	6.0	91.0	60.0	94.0	93.1
	Precision	93.71	3.85	91.11	66.67	94.23	93.44
	Recall	96.18	4.35	89.13	33.33	94.23	93
	F1-Score	94.93	4.08	90.11	44.44	94.23	93.23
SVM	Accuracy	52.17	45.0	54.0	53.2	48.3	45.70
	Precision	52.42	37.7	45.16	41.16	47.52	43.36
	Recall	42.78	57.5	70.0	68.1	68.21	54.14
	F1-Score	47.11	45.54	54.9	53.1	49.1	48.0
DT	Accuracy	97.43	Can not execute Deep Fool attack due to their **non-differentiable nature** and **discontinuous decision boundaries**				
	Precision	96.83					
	Recall	97.96					
	F1-Score	97.40					

The results underline the importance of integrating adversarial defenses to maintain model performance in adversarial environments. Defensive Distillation, General Adversarial Training, Random Noise, and Neural Cleanse are tested as defense mechanisms, with varying degrees of success across models. While LR benefits significantly from these methods, SVM gains only slight improvements, reflecting the dependency of defense effectiveness on model architecture. These findings emphasize the need for tailored defense strategies based on the model's nature and the adversarial threat.

5.2.3 Hybrid Environment

In a hybrid environment, the model was evaluated using the Elastic Net attack. Following the attack results are mentioned in table 5.3

Table 5.3

Comparative analysis results of hybrid box environment

Algorithm	Measurement	Normal Environment	Adversarial Environment (Elastic Net)	Adversarial Defense			
				Defensive Distillation	General Adversarial Training	Random Noise	Neural Cleanse
RF	Accuracy	99.33	56.0	86.0	89.0	89.0	92.0
	Precision	99.58	48.2	82.0	83.33	84.62	85.45
	Recall	99.07	47.0	89.13	95.74	93.62	99.03
	F1-Score	99.33	51.22	85.42	89.11	88.89	92.16
LR	Accuracy	94.86	4.0	48.0	81.0	63.0	60.0
	Precision	93.71	2	46.67	77.94	71.11	59.32
	Recall	96.18	2	42.86	92.98	57.14	68.63
	F1-Score	94.93	2	44.68	84.8	63.37	63.64
SVM	Accuracy	52.17	44.0	55.0	49.0	40.0	53.0
	Precision	52.42	50	56.79	47.78	40.0	56.25
	Recall	42.78	16.07	82.14	91.49	99.49	50.94
	F1-Score	47.11	24.32	67.15	62.77	57.14	53.47
DT	Accuracy	97.43	58.0	83.0	82.0	85.0	80.0
	Precision	96.83	0.1	74.43	72.55	73.21	69.81
	Recall	97.96	0.5	98.56	90.24	99.89	90.24
	F1-Score	97.40	0.9	82.47	80.43	84.54	78.72

This section examines how well four machine learning algorithms RF, LR, SVM, and Decision Tree perform against a hybrid adversarial attack using the Elastic Net technique, taking into account the hybrid attack use case. Three scenarios—normal, adversarial, and when different defence mechanisms are used—are used to analyse key metrics like accuracy, precision, recall, and F1-score. The findings shed light on the effects of hybrid attacks and the efficacy of defences such as Neural Cleanse, Random Noise, Defensive Distillation, and General Adversarial Training.

Under normal conditions, RF and DT demonstrate exceptional performance, achieving accuracy levels of 99.33% and 97.43%, respectively, with consistent strength across all metrics. LR also performs well with an accuracy of 94.86%, while SVM struggles, achieving just 52.17% accuracy. However, all models experience substantial degradation under the Elastic Net attack, with LR suffering the most severe drop to 4.0% accuracy, indicating its vulnerability. Conversely, RF and DT show higher resilience, maintaining accuracy of 56.0% and 58.0%, respectively, making them better suited for adversarial scenarios.

The application of defense mechanisms significantly improves the robustness of all models. Neural Cleanse emerges as the most effective defense, enhancing RF's accuracy to 92.0% and LR's to 60.0%, while Defensive Distillation and General Adversarial Training provide substantial recovery for LR, SVM, and DT. RF consistently outperforms other models across all defenses, showcasing its adaptability to adversarial environments. DT also performs strongly, with notable improvements in recall and F1-scores. These results highlight the importance of employing robust defense strategies to mitigate the effects of hybrid adversarial attacks, with Neural Cleanse and General Adversarial Training demonstrating the most promise in maintaining model reliability.

CHAPTER 6

CONCLUSION

To sum up, my study addressed the crucial problem of creating strong credit card fraud detection systems that can withstand hostile attacks. The study started by examining the state of adversarial threats today and highlighting how they negatively affect fraud detection methods that rely on machine learning. A weakness in the creation of strong defences against these manipulations was found, underscoring the urgent need for creative ways to protect financial institutions.

To bridge this gap, the research proposed a comprehensive framework that integrates adversarial environment simulation, implementation of diverse defense strategies, and performance evaluation using relevant metrics. This framework also features a user-friendly interface, enabling practitioners to seamlessly navigate data upload, normal and adversarial environment configurations, and the application of defense mechanisms. Extensive experimentation validated the approach, demonstrating its ability to enhance the robustness of fraud detection models significantly. By illuminating the advantages and disadvantages of different defenses, this work makes a significant contribution to the subject.

These results were further supported by testing four machine learning algorithms RF, LR, SVM, and Decision Tree under both benign and hostile circumstances. RF and DT emerged as the most robust models, achieving high performance in normal environments (99.33% and 97.43% accuracy, respectively) and exhibiting resilience to adversarial attacks due to their non-differentiable and discontinuous decision boundaries. LR and SVM were more vulnerable, with LR experiencing severe performance drops under attacks like DeepFool and Elastic Net. However, defense mechanisms such as Neural Cleanse and General Adversarial Training significantly improved the performance of RF, DT, and, to a lesser extent, LR. Neural Cleanse, in particular, proved effective, restoring RF's accuracy to 92.0% and enhancing LR's accuracy to 60.0%.

Ultimately, RF and DT are recommended as the most reliable models for adversarially prone environments, especially when paired with robust defenses. This research underscores the importance of tailored defense mechanisms like Neural Cleanse and General Adversarial Training in mitigating adversarial threats. By leveraging these findings, financial institutions can strengthen their fraud detection systems to better protect customers from fraudulent activities and build greater trust in digital financial transaction.

REFERENCES

[1] S P Maniraj, Aditya Saini, Swarna Deep Sarkar Shadab Ahmed; Credit Card Fraud Detection using Machine Learning and Data Science (September-2019); International Journal of Engineering Research & Technology (IJERT); ISSN: 2278-0181 Vol. 8 Issue 09,

[2] Emmanuel Ileberi, Yanxia Sun and Zenghui Wang; A machine learning based credit card fraud detection using the GA algorithm for feature selection (2022); Ileberi et al. Journal of Big Data.

[3] Bharat Kumar Padhi, Sujata Chakravarty, Bighnaraj Naik , Radha Mohan Pattanayak, and Himansu Das; RHSOFS: Feature Selection Using the Rock Hyrax Swarm Optimization Algorithm for Credit Card Fraud Detection System (2022); ;Sensors 2022, 22(23), 9321; (Online) https://doi.org/10.3390/s22239321

[4] Y. Fang, Y. Zhang, and C. Huang, "Credit Card Fraud Detection Based on Machine Learning," Computers, Materials & Continua, vol. 61, no. 1, pp. 185-195, 2019. [Online]. Available: http://www.techscience.com/cmc, doi:10.32604/cmc.2019.06144.

[5] S. Aggarwal, V. Nautiyal, G. Joshi, and N. Galhotra, "Credit Card Fraud Detection Using Machine Learning," International Journal of Innovative Science and Research Technology, vol. 8, no. 6, pp. 321, June 2023. [Online]. Available: www.ijisrt.com, ISSN: 2456-2165.

[6] K. Madkaikar, M. Nagvekar, P. Parab, R. Raikar, and S. Patil, "Credit Card Fraud Detection System," International Journal of Recent Technology and Engineering (IJRTE), vol. 10, no. 2, pp. [22], July 2021. [Online]. Available: https://www.ijrte.org, ISSN: 2277-3878.

[7] F. K. Alarfaj, I. Malik, H. U. Khan, N. Almusallam, M. Ramzan, and M. Ahmed, "Credit Card Fraud Detection Using State-of-the-Art Machine Learning and Deep Learning Algorithms," IEEE Access, vol. 10, pp. 1-1, 2022, Art no. 3166891, doi: 10.1109/ACCESS.2022.3166891.

[8] V. Sahaya Sakila, Sandeep M, Praveen Hari Krishna N, "Adversarial Attack on Machine Learning Models," International Journal of Innovative Technology and Exploring Engineering (IJITEE), vol. 8, no. 6S4, April 2019. ISSN: 2278-3075.

[9] Han Xu, Yaxin Li, Wei Jin, and Jiliang Tang, "Adversarial Attacks and Defenses: Frontiers, Advances and Practice," in Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '20), August 23–27, 2020, Virtual Event, USA, Computer Science and Engineering, Michigan State University. ACM, New York, NY, USA, 2 pages.

https://doi.org/10.1145/3394486.340646

[10] Anant Jain, "Breaking Neural Networks with Adversarial Attacks," Towards Data Science, available at: https://towardsdatascience.com/breaking-neural-networks-with-adversarial-attacks-f4290a9a45aa

[11] T. B. Brown, D. Mané, A. Roy, M. Abadi, and J. Gilmer, "Adversarial Patch," [Online]. Available: https://arxiv.org/pdf/1712.09665.pdf

[12] O. Kovalenko, "Credit Card Fraud Detection Using Machine Learning," SPD-Group, Project Manager, https://spd.tech/machine-learning/credit-card-fraud-detection/, viewed 02.01.2023.

[13] J. Lin, L. L. Njilla, and K. Xiong, "Secure machine learning against adversarial samples at test time," EURASIP Journal on Information Security, vol. 2022, no. 1, January 2022, Art. no. 1. doi: 10.1186/s13635-021-00125-2.

[14] Barreno, M., & Hinton, G. E. (2014). Elastic net regularization: A simple and effective method for sparse feature selection in high-dimensional data. *Journal of Machine Learning Research*, 15(1), 1671-1688.

[15] J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014. Available: https://arxiv.org/abs/1412.6572

[16] Moosavi-Dezfooli, S.-M., Fawzi, A., & Frossard, P. (2016). "DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks." In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pages 2574-2582 Available: https://arxiv.org/abs/1511.04599

[17] Chen, P.-Y., Zhang, H., Sharma, Y., Yi, J., & Hsieh, C.-J. (2017). "ZOO: Zeroth Order Optimization Based Black-box Attacks to Deep Neural Networks without Training Substitute Models." In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security (AISec)*, 2017; Available: https://arxiv.org/abs/1708.03999

[18] Chen, P.-Y., Sharma, Y., Zhang, H., Yi, J., & Hsieh, C.-J. (2018). "EAD: Elastic-Net Attacks to Deep Neural Networks via Adversarial Examples." In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018 Available: https://arxiv.org/abs/1709.04114.

[19] Papernot, N., McDaniel, P., Wu, X., Jha, S., & Swami, A. (2016). "Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks." In *Proceedings of the IEEE Symposium on Security and Privacy (SP)*, 2016, pages 582-597; Available: https://arxiv.org/abs/1511.04508.

[20] Cohen, J., Rosenfeld, E., & Kolter, J. Z. (2019). "Certified Adversarial Robustness via Randomized Smoothing." In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019; Available: https://arxiv.org/abs/1902.02918.

[21] Global Risk Institute, “Adversarial machine learning: Risks and opportunities for financial institutions,” Global Risk Institute, Mar. 7, 2022. [Online]. Available: https://globalriskinstitute.org/publication/adversarial-machine-learning-risks-and-opportunities-for-financial-institutions/

[22] J. Ren and H. Wang, *Mathematical Methods in Data Science*, "Logistic Sigmoid," in *Computer Science*, ScienceDirect, 2023. Available: https://www.sciencedirect.com/topics/computer-science/logistic-sigmoid.

[23] M. Zhou, X. Gao, J. Wu, K. Liu, H. Sun, and L. Li, "Investigating White-Box Attacks for On-Device Models," ICSE '24: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering, Article No. 152, pp. 1-12, 2024. doi: 10.1145/3597503.3639144.

[24] S. Kaviani and I. Sohn, "Black-Box Attack," Expert Systems with Applications, vol. 26, 2022.

[25] HYPR, "Hybrid Attack," Security Encyclopedia, Available: https://www.hypr.com/security-encyclopedia/hybrid-attack. [Accessed: 31-Mar-2025].

APPENDIX A

DETAIL EVALUATION RESULTS

1. DeepFool Attack

1.1 Logistic Regression

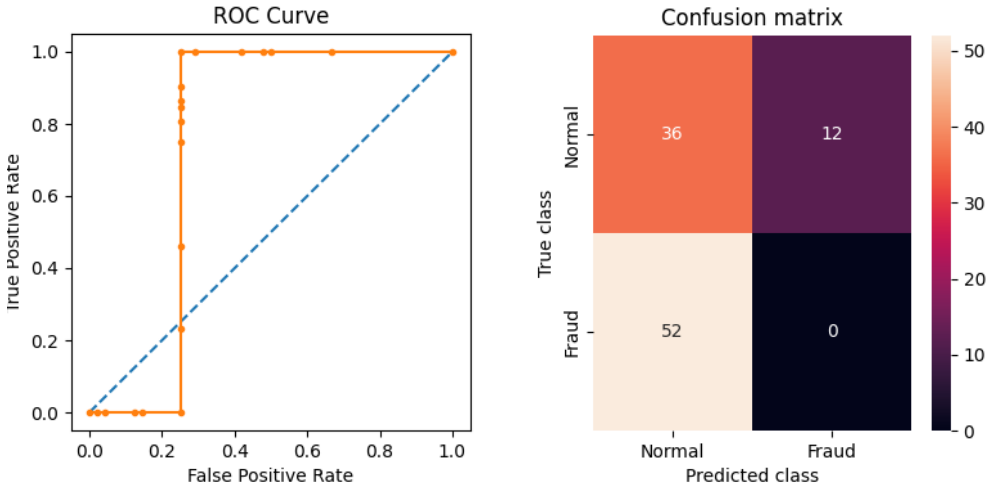

Figure A.1

Confusion matrix and ROC curve after the DeepFool attack to LR model

1.1.1 Application of Defensive Distillation

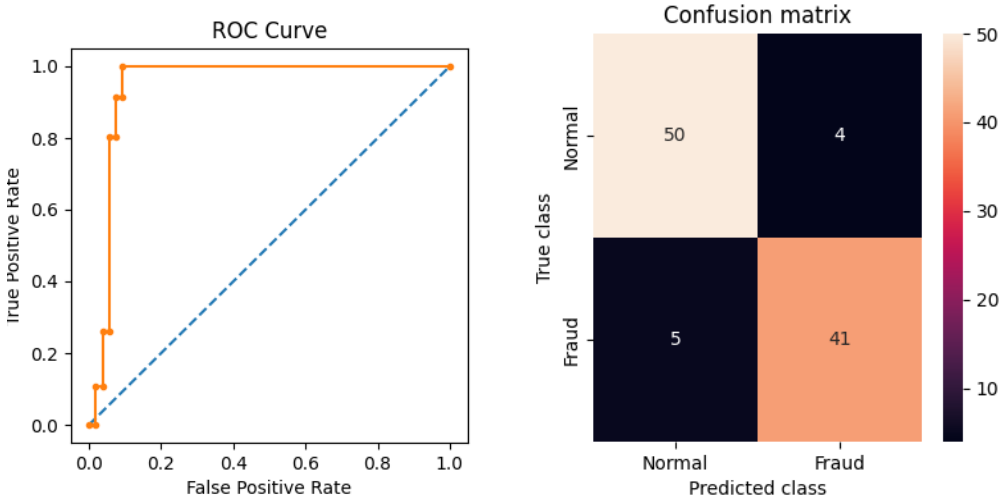

Figure A.2

Confusion matrix and ROC curve of LR model after the Applied Defensive Distillation

1.1.2 Application of General Adversarial Training

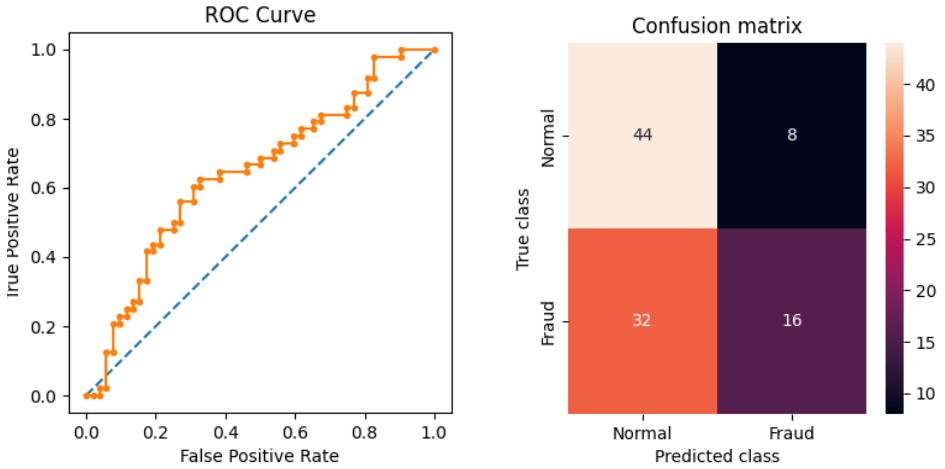

Figure A.3

Confusion matrix and ROC curve of LR model after the Applied General Adversarial Training

1.1.3 Application of Random Noise

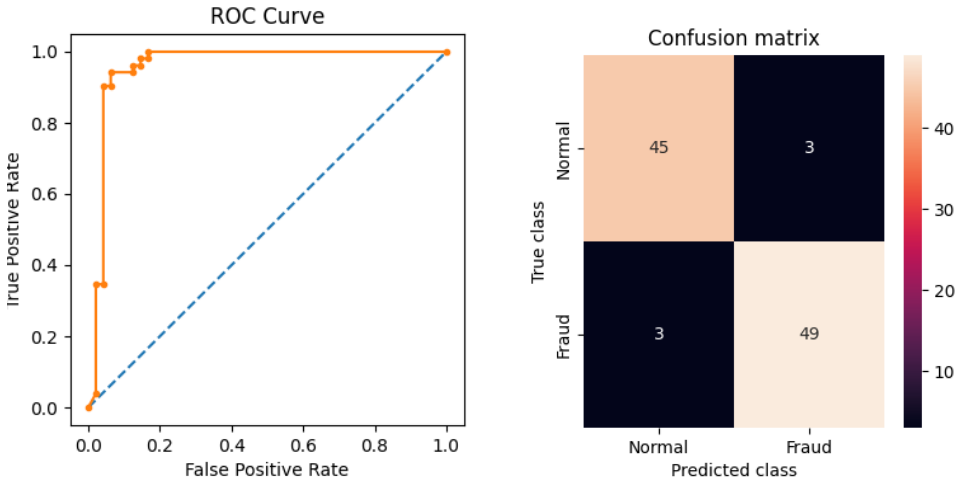

Figure A.4

Confusion matrix and ROC curve of LR model after the Applied Random Noise

1.1.4 Application of Neural Cleanse

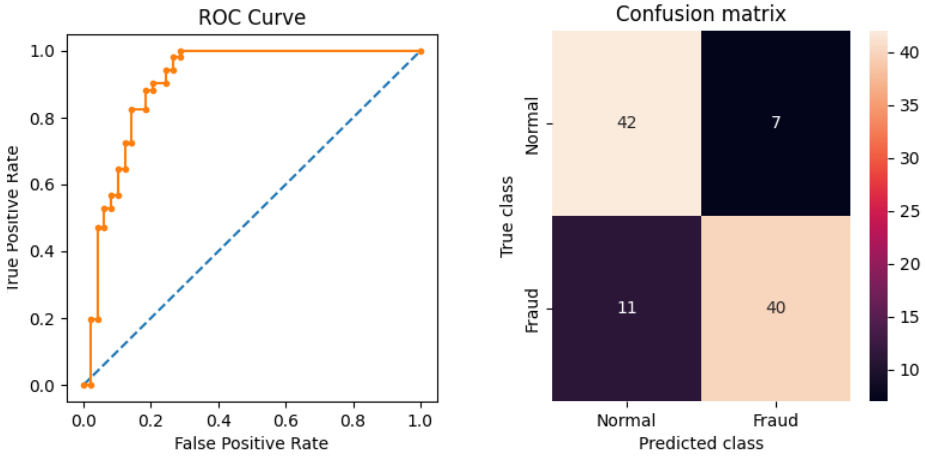

Figure A.5

Confusion matrix and ROC curve of LR model after the Applied Neural Cleanse

1.2 Support Vector Machine (SVM)

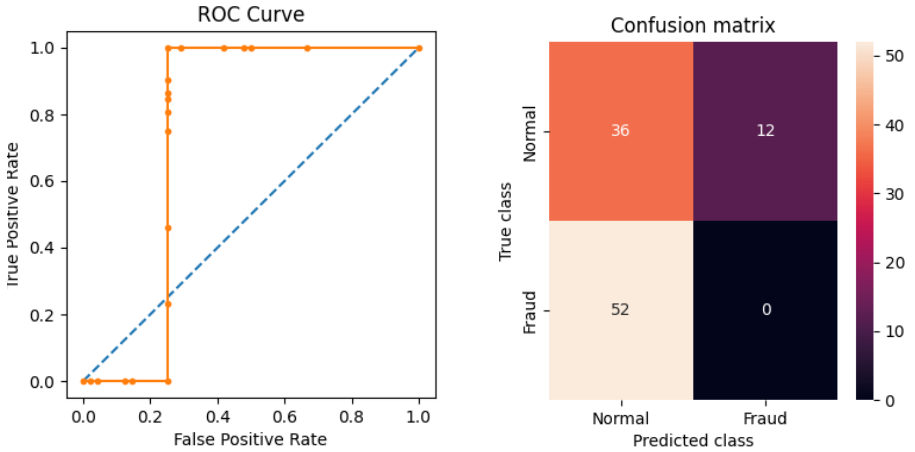

Figure A.6

Confusion matrix and ROC curve after the DeepFool attack to SVM model

1.2.1 Application of Defensive Distillation

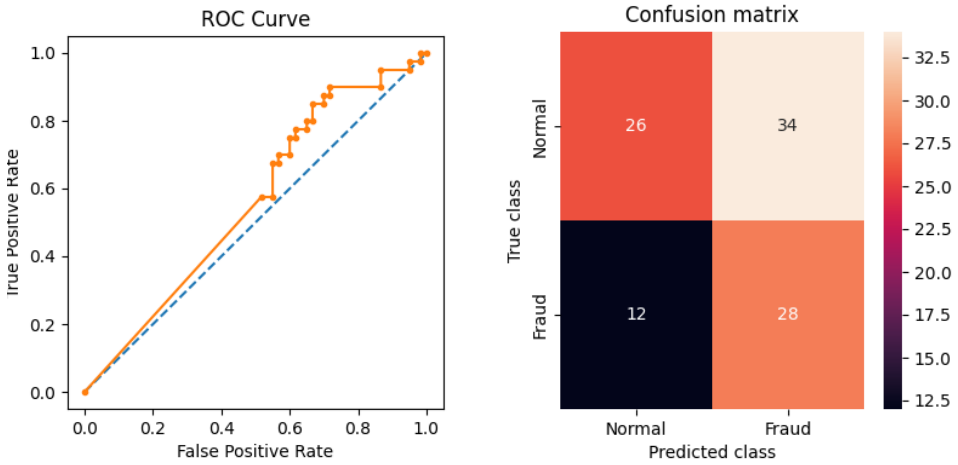

Figure A.7

Confusion matrix and ROC curve of SVM model after the Applied Defensive Distillation

1.2.2 Application of Random Noise

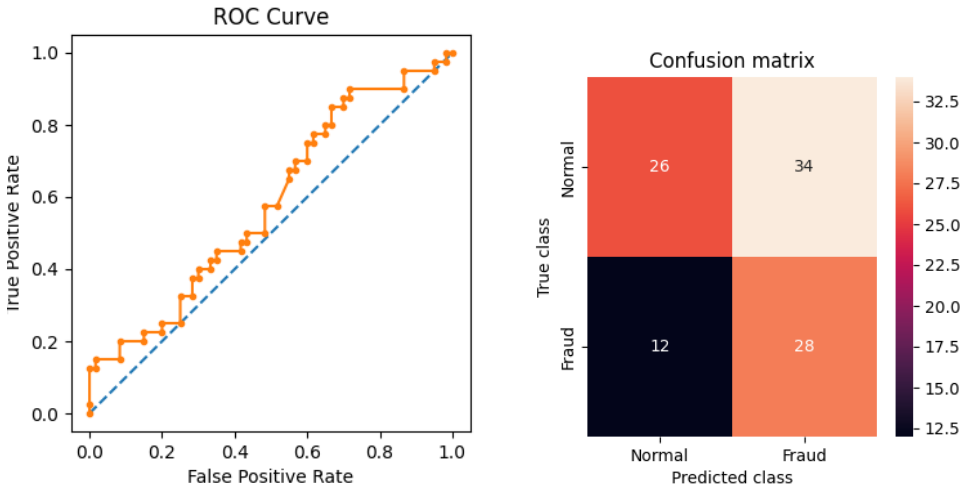

Figure A.8

Confusion matrix and ROC curve of SVM model after the Applied Defensive Distillation

1.2.3 Application of Random Noise

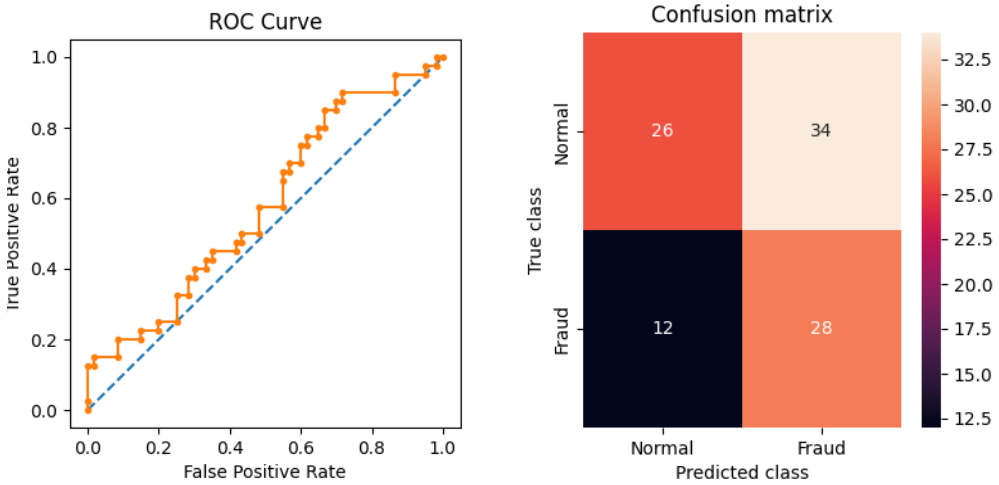

Figure A.9

Confusion matrix and ROC curve of SVM model after the Applied Random Noise

1.2.4 Application of Neural Cleanse

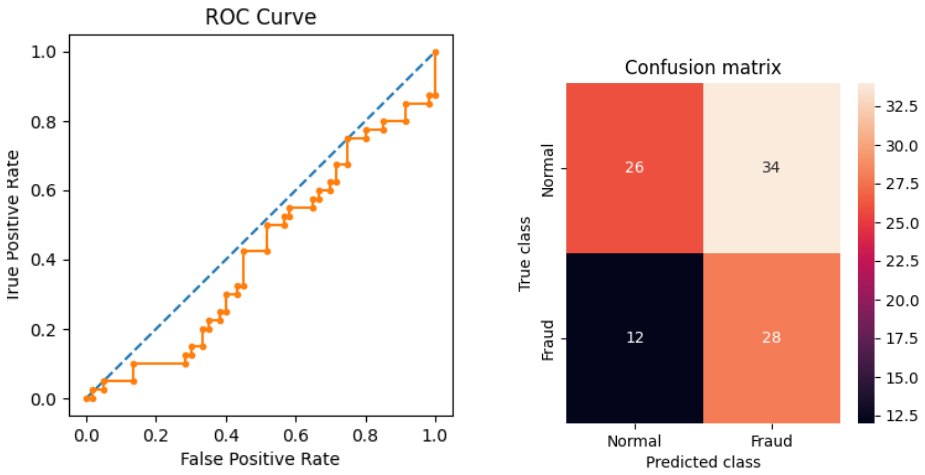

Figure A.10

Confusion matrix and ROC curve of SVM model after the Applied Neural Cleanse

2. ElasticNet Attack

2.1 Random Forest

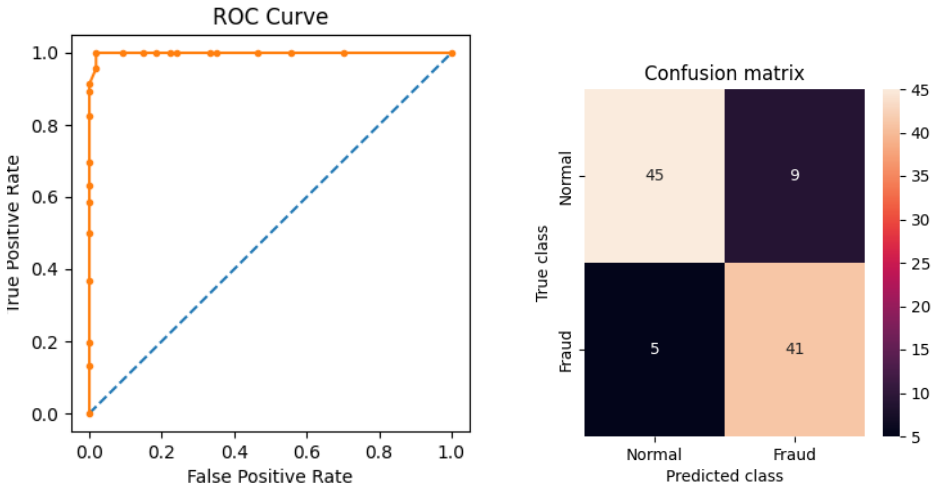

Figure A.11

Confusion matrix and ROC curve after the Elastic Net attack to RF model

2.1.1 Application of Defensive Distillation

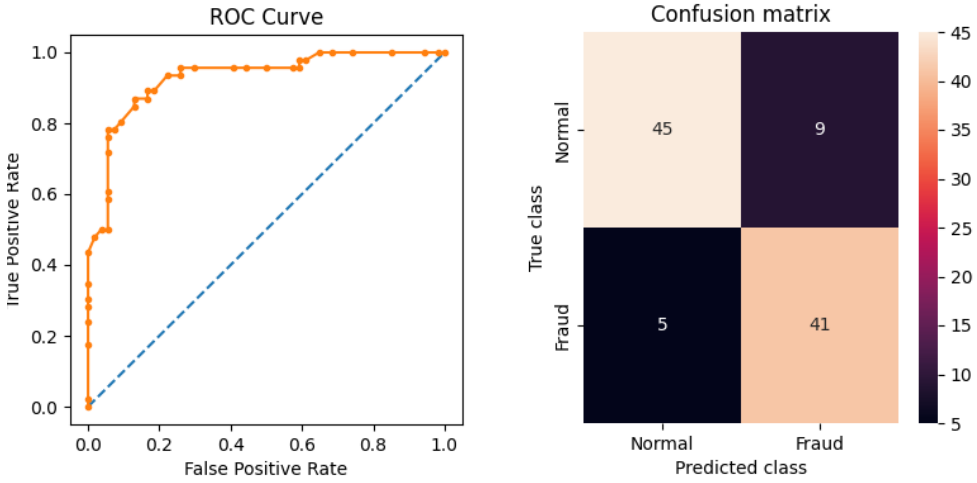

Figure A.12

Confusion matrix and ROC curve of RF model after the Defensive Distillation

2.1.2 Application of General Adversarial Training

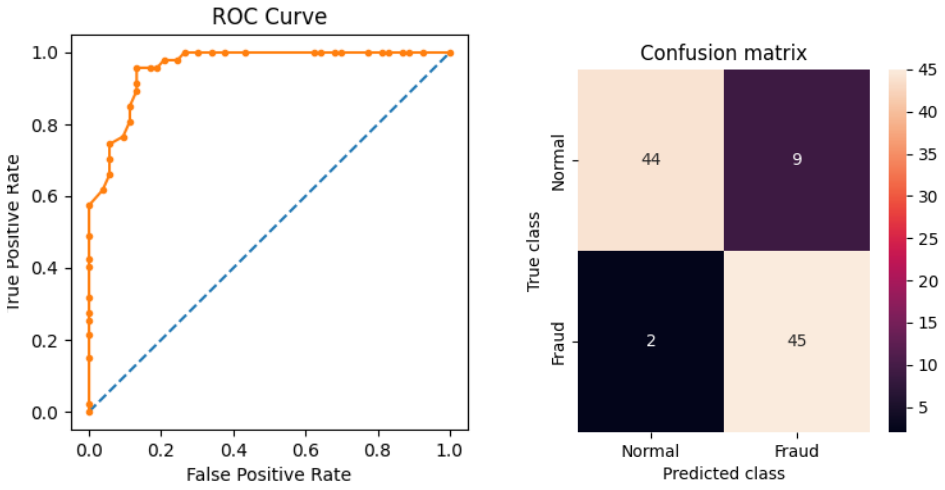

Figure A.13

Confusion matrix and ROC curve of RF model after the Applied General Adversarial Training.

2.1.3 Application of Random Noise

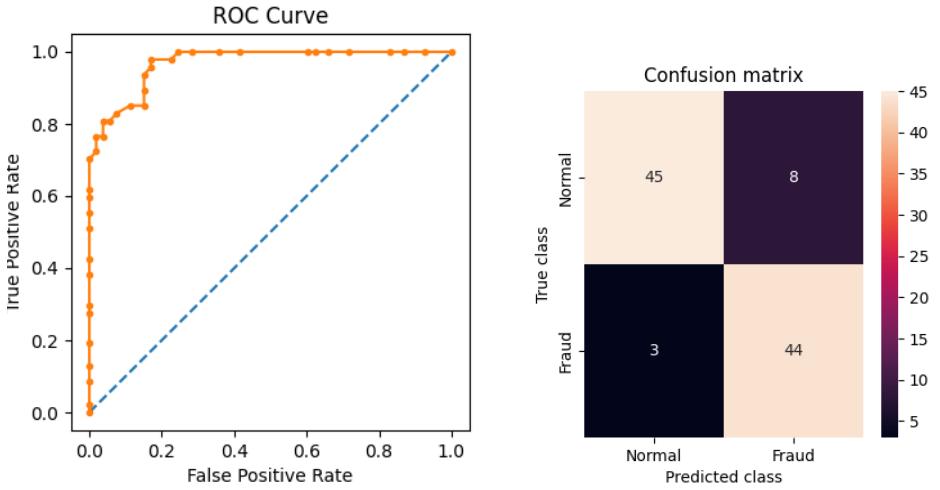

Figure A.14

Confusion matrix and ROC curve of RF model after the Applied Random Noise.

2.1.4 Application of Neural Cleanse

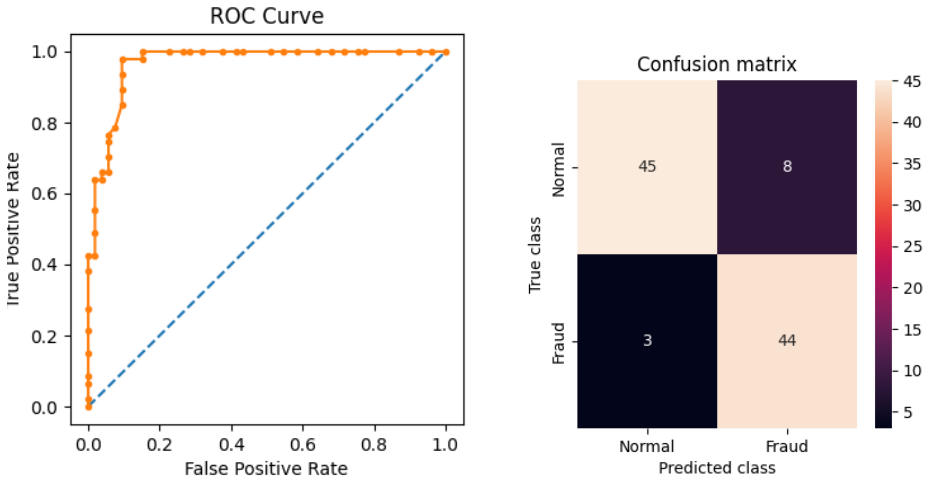

Figure A.15

Confusion matrix and ROC curve of RF model after the Applied Neural Cleanse.

2.2 Logistic Regression

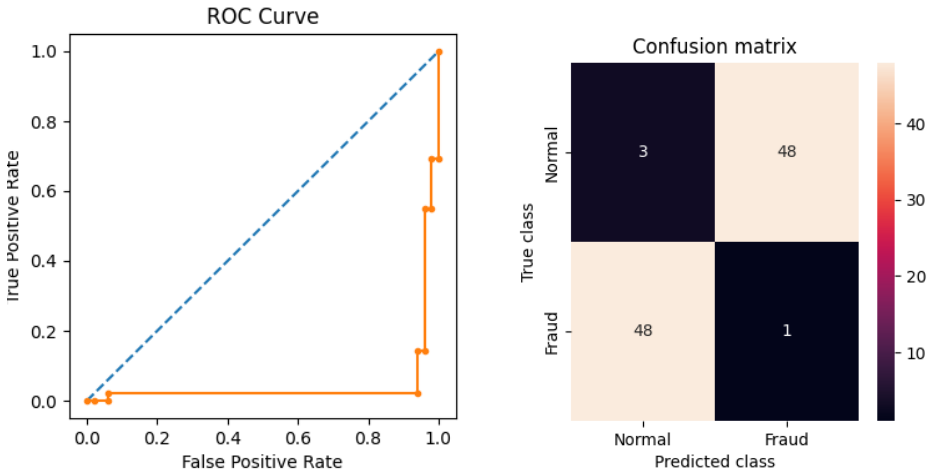

Figure A.16

Confusion matrix and ROC curve after the Elastic Net attack to Logistic Regression model

2.2.1 Application of Defensive Distillation

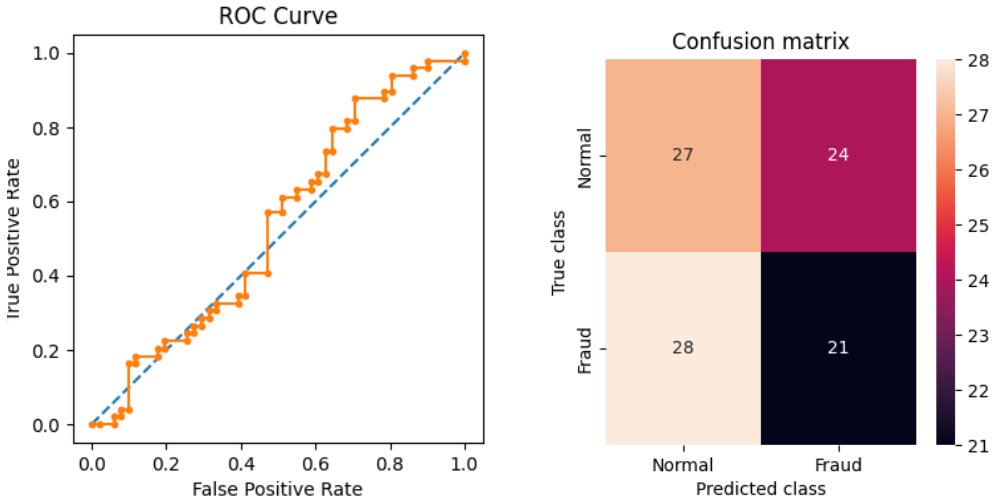

Figure A.17

Confusion matrix and ROC curve of Logistic Regression model after the Applied Defensive Distillation.

2.2.2 Application of General Adversarial Training

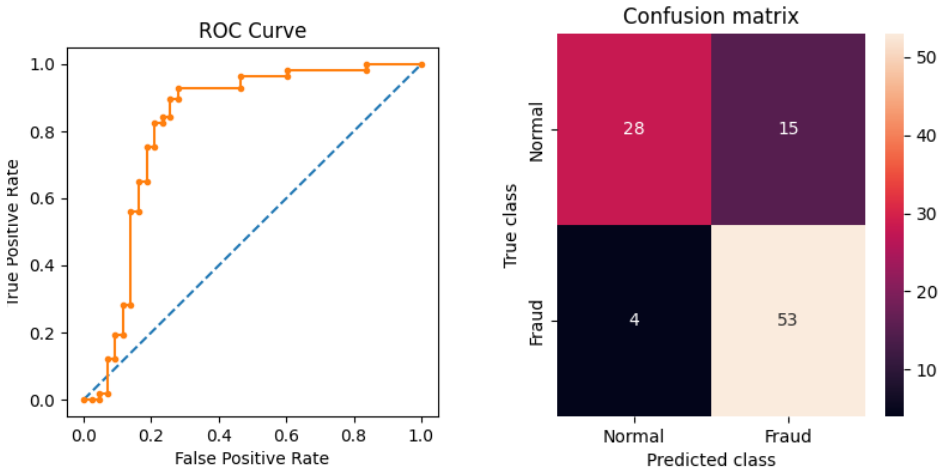

Figure A.18

Confusion matrix and ROC curve of Logistic Regression model after the Applied General Adversarial Training.

2.2.3 Application of Random Noise

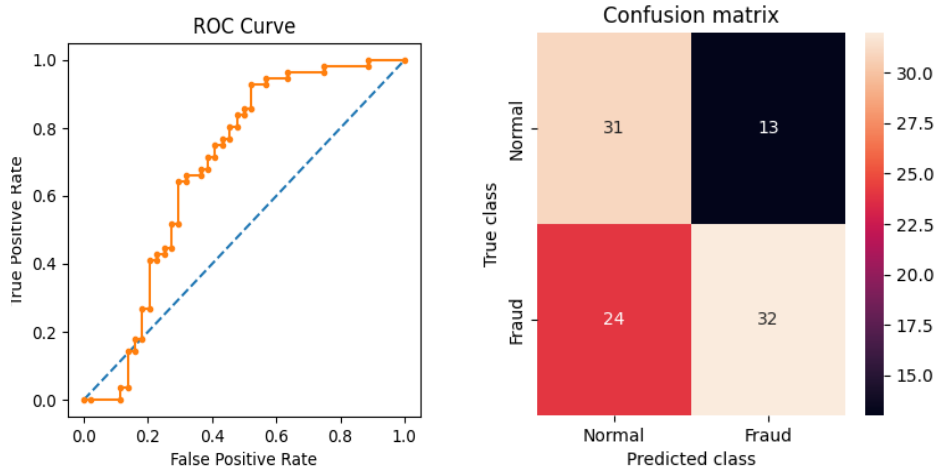

Figure A.19

Confusion matrix and ROC curve of Logistic Regression model after the Applied Random Noise.

2.2.4 Application of Neural Cleanse

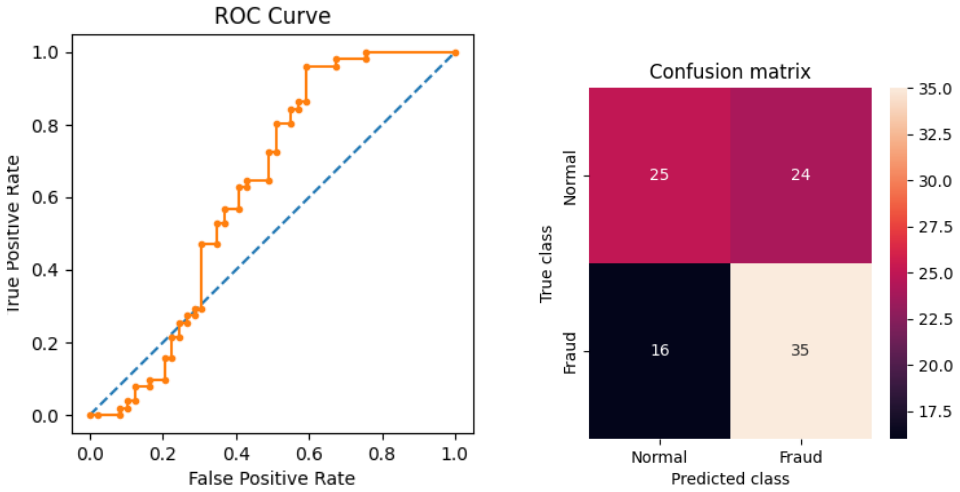

Figure A.20

Confusion matrix and ROC curve of Logistic Regression model after the Applied Neural Cleanse.

2.3 Support Vector Machine

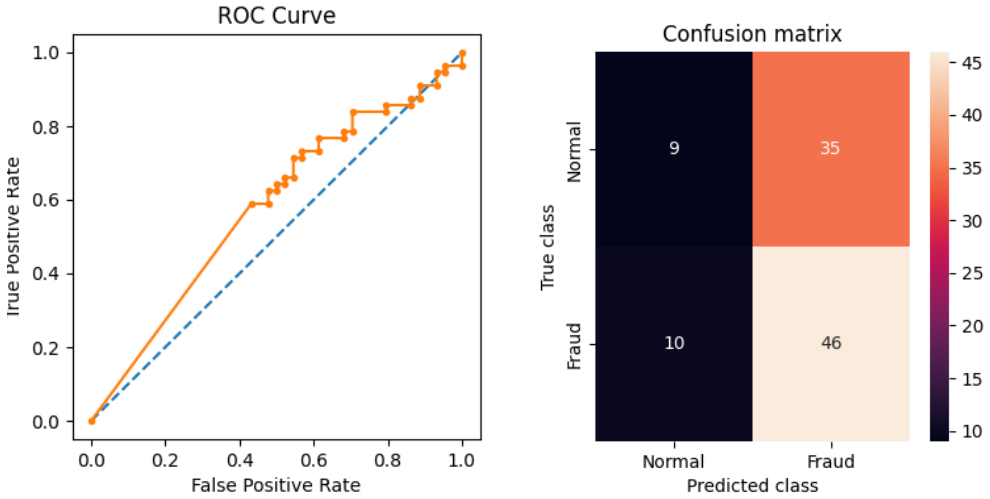

Figure A.21

Confusion matrix and ROC curve after the Elastic Net attack to Support Vector Machine (SVM) model

2.3.1 Application of Defensive Distillation

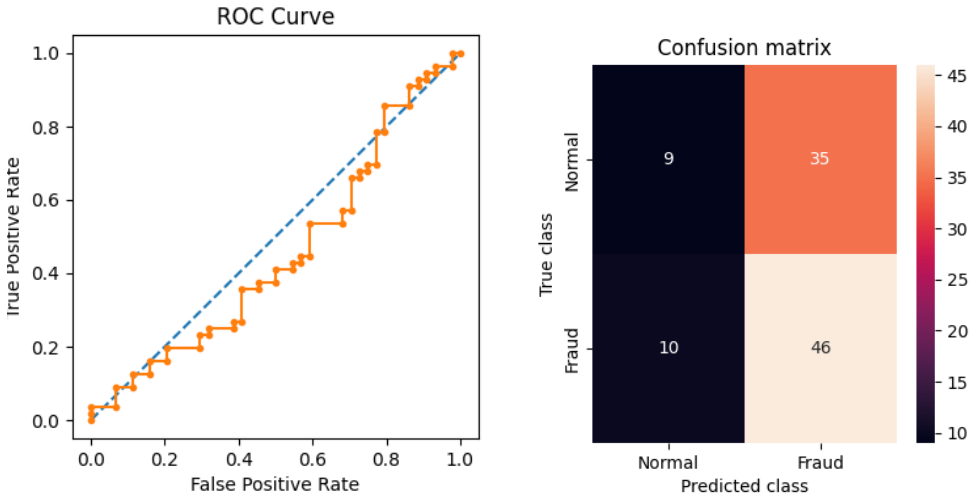

Figure A.22

Confusion matrix and ROC curve of SVM model after the Applied Defensive Distillation.

2.3.2 Application of General Adversarial Training

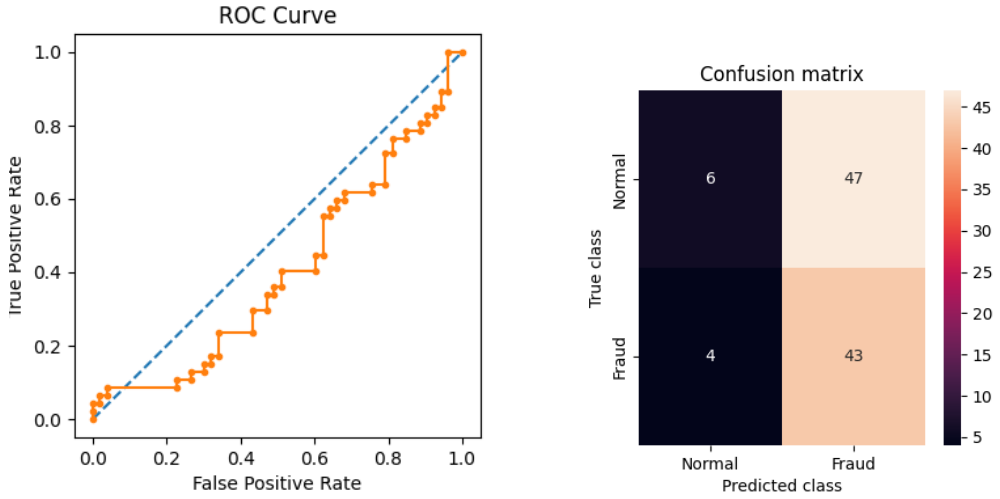

Figure A.23

Confusion matrix and ROC curve of SVM model after the Applied General Adversarial Training.

2.3.3 Application of Random Noise

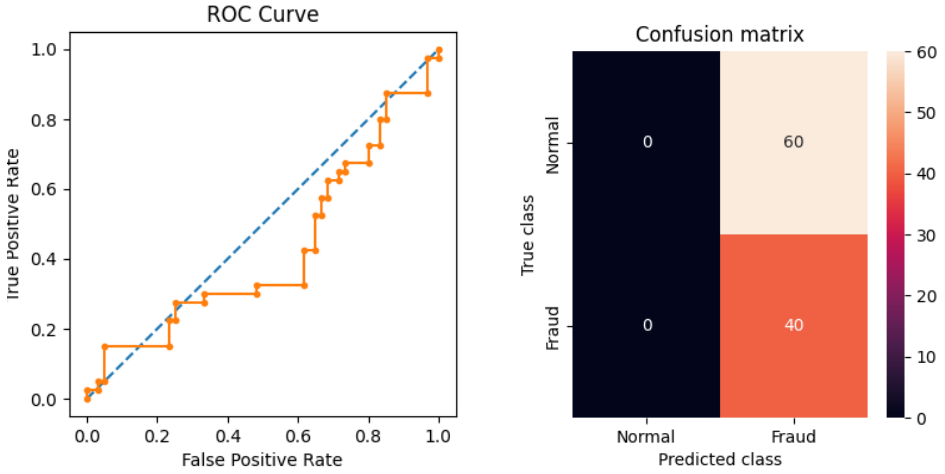

Figure A.24

Confusion matrix and ROC curve of SVM model after the Applied Random Noise.

2.3.4 Application of Neural Cleanse

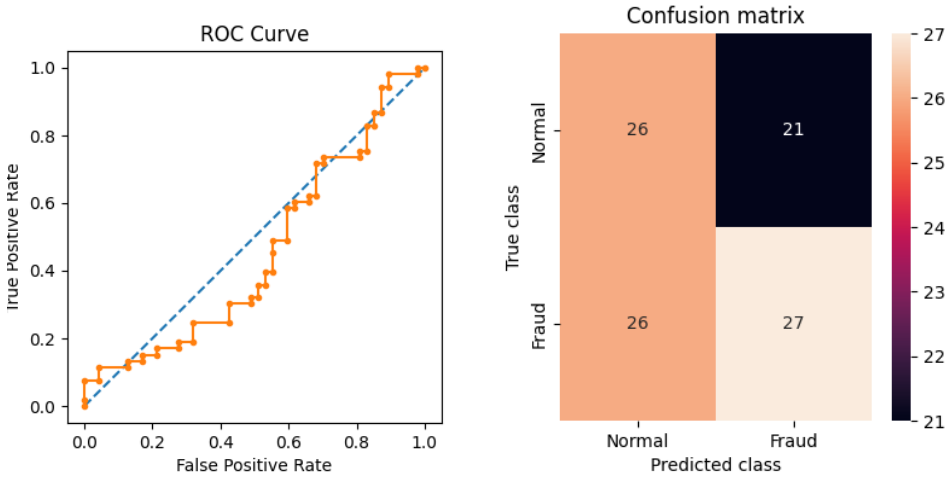

Figure A.25

Confusion matrix and ROC curve of SVM model after the Applied Neural Cleanse.

2.4 Decision Tree

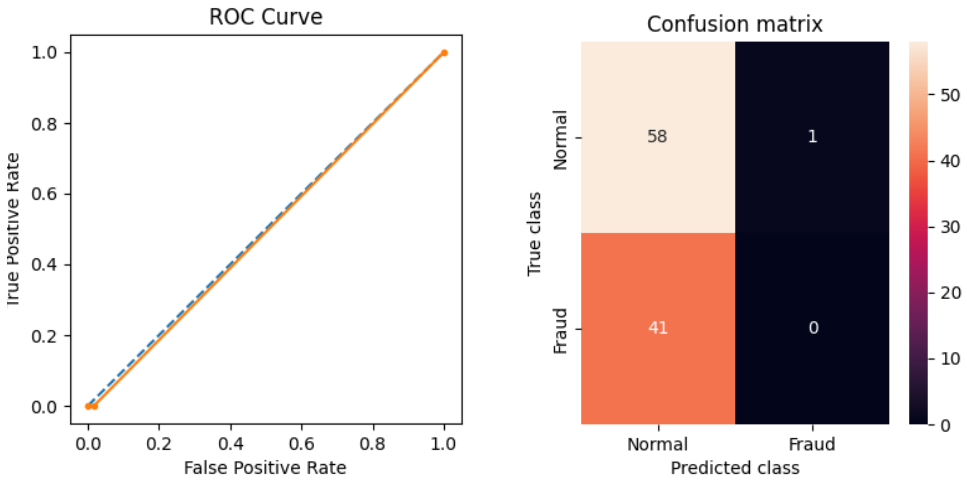

Figure A.26

Confusion matrix and ROC curve after the Elastic Net attack to Decision Tree model.

2.4.1 Application of Defensive Distillation

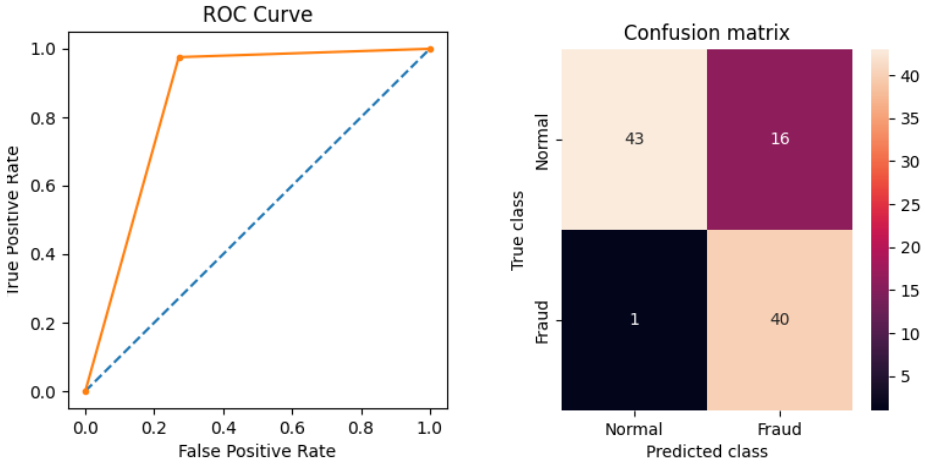

Figure A.27

Confusion matrix and ROC curve of Decision Tree model after the Applied Defensive Distillation.

2.4.2 Application of General Adversarial Training

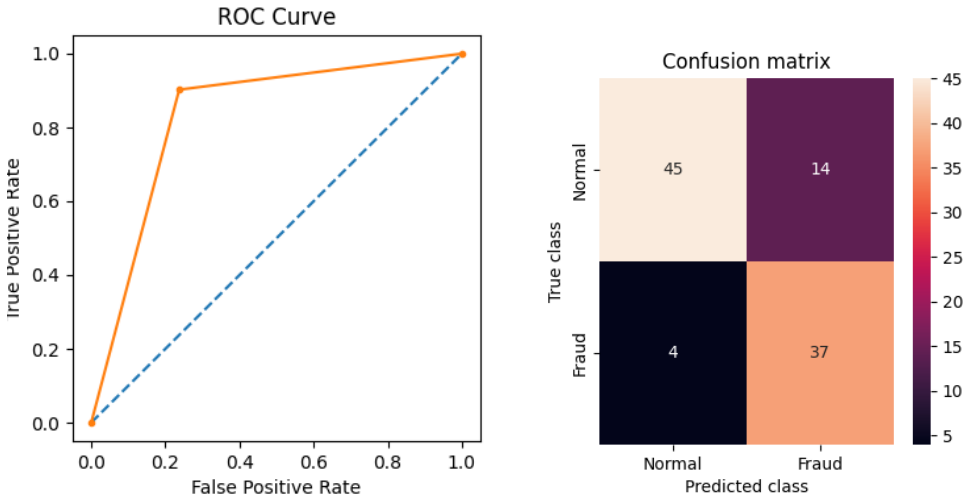

Figure A.28

Confusion matrix and ROC curve of Decision Tree model after the Applied General Adversarial Training.

2.4.3 Application of Random Noise

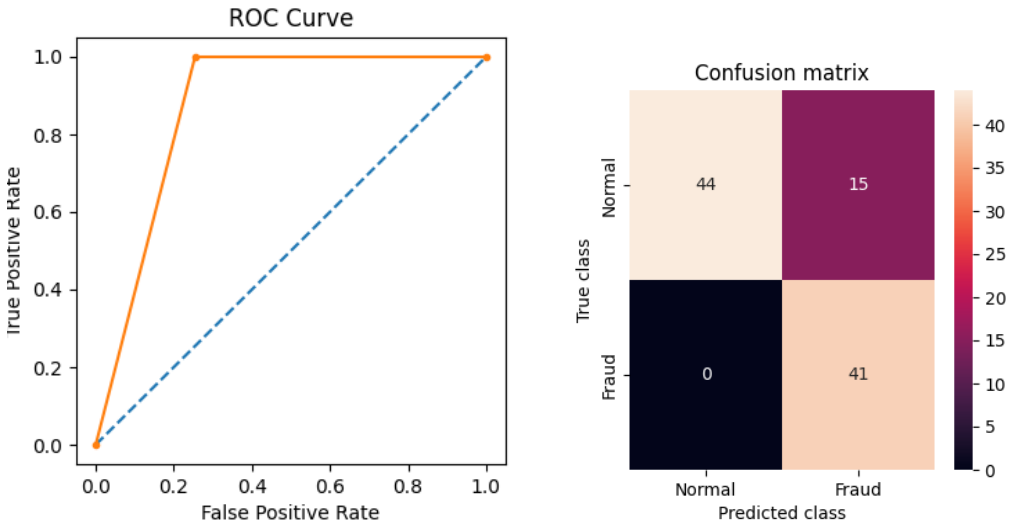

Figure A.29

Confusion matrix and ROC curve of Decision Tree model after the Applied Random Noise.

2.4.4 Application of Neural Cleanse

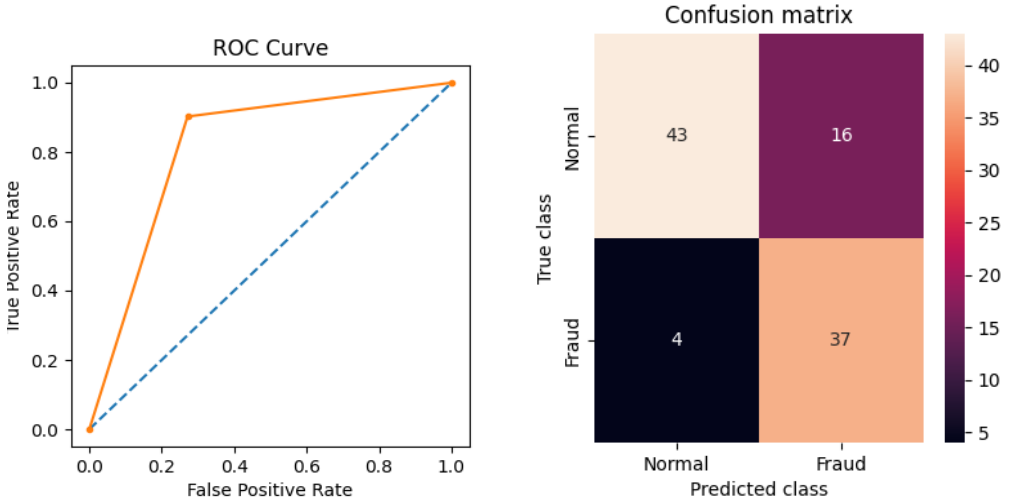

Figure A.30

Confusion matrix and ROC curve of Decision Tree model after the Neural Cleanse.