TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Introduction

Chapter 1

INTRODUCTION

1.1 Overview

Credit card fraud has become a significant challenge in the digital era, impacting financial systems and consumer trust worldwide. The increasing reliance on online transactions necessitates advanced solutions to detect and prevent fraudulent activities. This project focuses on building a robust credit card fraud detection system utilizing machine learning techniques. By analyzing historical transaction data, the system aims to identify fraudulent patterns and distinguish them from legitimate transactions with high accuracy. The proposed system addresses the limitations of traditional fraud detection methods and adapts to evolving fraud techniques.

1.2 Purpose, Scope, and Applicability

Purpose: The primary goal of this project is to create a robust and efficient machine learning-based system for detecting credit card fraud. The system is designed to enhance the accuracy and reliability of fraud detection processes, addressing the challenges faced by traditional methods.

Scope: This project aims to develop a scalable solution capable of handling vast amounts of transaction data while adapting to various fraud scenarios. By utilizing modern algorithms and data preprocessing techniques, the system ensures real-time detection of fraudulent activities with a high degree of precision. It is designed to evolve with changing patterns in fraudulent behavior, ensuring continued effectiveness over time.

Applicability: The system is particularly beneficial for financial institutions, payment gateways, and e- commerce platforms. It provides these entities with a powerful tool to enhance transaction security, protect against financial losses, and build trust among customers by reducing the risks associated with fraudulent transactions.

1.3 Literature Review

Fraud detection has traditionally relied on rule-based systems, manual reviews, and statistical methods. Rule- based systems, while simple to implement, often result in high false positive rates and struggle to adapt to new fraud patterns. Manual reviews are labor-intensive and not scalable for large datasets. Statistical methods, such as clustering and regression, depend on assumptions about data distribution, limiting their effectiveness.

Recent advancements in machine learning have demonstrated significant improvements in fraud detection. Algorithms like Random Forest, XGBoost, and Artificial Neural Networks (ANNs) provide enhanced accuracy by analyzing complex data patterns. Techniques like Synthetic Minority Oversampling Technique (SMOTE) address data imbalance issues, enabling better training of machine learning models. These developments form the foundation of the proposed system.

These methods face challenges such as high false positive rates, lack of scalability, and poor adaptability to evolving fraud tactics. Addressing these gaps is crucial for developing a reliable fraud detection system.

1.3.1 Existing Systems and their Drawbacks

Traditional credit card fraud detection systems have been used for years to combat fraudulent activities. While they have played a crucial role in the past, they come with significant limitations that reduce their effectiveness in the modern digital landscape. Below is an explanation of these systems and their drawbacks:

1. Rule-Based Systems

Rule-based systems rely on predefined rules and thresholds to detect potentially fraudulent transactions. For example, a rule might flag transactions exceeding a specific monetary value or originating from unfamiliar geographic locations. These systems are simple to implement and provide quick results. However, they suffer from several drawbacks:

- **High Maintenance Requirements**: Rules must be continuously updated to address emerging fraud tactics. This constant need for adjustment makes the system resource-intensive and time-consuming to maintain.
- **Inability to Detect New Fraud Patterns**: Fraudsters often develop methods that bypass existing rules. Since rule-based systems lack adaptability, they struggle to identify novel patterns, making them ineffective against evolving fraud strategies.
- **High False Positive Rates**: Legitimate transactions that meet certain criteria (e.g., a large purchase in a new location) may be wrongly flagged as fraudulent, causing inconvenience to customers and damaging their trust in the system.

2. Manual Reviews

Manual reviews involve human analysts examining flagged transactions to determine their legitimacy. While this approach benefits from human judgment and expertise, it has several limitations:

- **Labor-Intensive**: Reviewing large volumes of flagged transactions is time-consuming and requires significant manpower, making it impractical for organizations handling thousands of daily transactions.
- **Lack of Scalability**: As transaction volumes grow, manual reviews become increasingly inefficient and unsustainable.
- **Subjectivity and Inconsistency**: Decisions made by human analysts can vary based on experience and judgment, leading to inconsistencies and errors in fraud detection.

3. Statistical Methods

Statistical methods use mathematical models such as regression and clustering to identify anomalies in transaction data. These techniques have been widely used in the past due to their ability to uncover patterns. However, they also have notable drawbacks:

- **Dependence on Assumptions**: Statistical methods often rely on assumptions about data distribution (e.g., normality or linearity). These assumptions may not hold true for complex and high-dimensional transaction data, limiting their effectiveness.

- **Limited Adaptability**: These methods are static and struggle to adjust to changing fraud patterns or datasets.
- **Performance Issues with Complex Data**: Modern financial systems generate diverse and intricate datasets that cannot be fully captured by traditional statistical models.

Challenges Across All Systems

Despite their differences, all these traditional approaches face common challenges:

- **High False Positive Rates**: Many legitimate transactions are mistakenly flagged, leading to unnecessary investigations and customer dissatisfaction.
- **Lack of Scalability**: These systems are often ill-equipped to handle the growing volume of transactions in today's digital economy.
- **Poor Adaptability**: Fraud tactics evolve rapidly, and traditional systems lack the flexibility to keep up with these changes.

The Need for Improvement

These limitations highlight the need for a more advanced and reliable fraud detection system. Modern machine learning-based approaches offer the adaptability, scalability, and precision required to address these challenges effectively. By leveraging data-driven techniques, the proposed system overcomes the shortcomings of traditional methods and provides a robust solution for fraud detection in the digital age.

1.4 Problem Statement

Design and implement a scalable and efficient machine learning-based system to detect fraudulent credit card transactions accurately. The system should leverage historical transaction data, handle class imbalances, and minimize false positives and negatives.

1.5 Objectives

The primary goal of this project is to develop an advanced credit card fraud detection system that leverages the power of machine learning to address the shortcomings of

traditional methods. The detailed objectives of the project are as follows:

1. Develop a Machine Learning-Based System

This objective focuses on designing and implementing multiple machine learning models for fraud detection. Models such as Artificial Neural Networks (ANNs), XGBoost, CatBoost, LightGBM, and Random Forest are utilized to ensure robust and reliable detection. Each of these models has unique strengths, with ANNs excelling at capturing complex patterns and relationships in the data, while tree-based models like XGBoost and Random Forest are highly efficient for tabular datasets with categorical and numerical features. By experimenting with multiple models, the system is designed to achieve optimal performance in identifying fraudulent transactions.

2. Minimize False Positives and Negatives

One of the critical challenges in fraud detection is balancing precision and recall to minimize false positives (legitimate transactions flagged as fraudulent) and false negatives (fraudulent transactions classified as legitimate). The system aims to achieve high detection accuracy without causing unnecessary disruptions to legitimate users. This objective ensures that the fraud detection system is reliable and user-friendly, maintaining trust while effectively combating fraudulent activities.

3. Address Data Imbalance

Fraud detection datasets are often highly imbalanced, with fraudulent transactions forming only a small percentage of the total data. This imbalance can lead to biased model training, where the algorithm focuses more on the majority class (legitimate transactions) and underperforms on the minority class (fraudulent transactions). To overcome this, the project incorporates the Synthetic Minority Oversampling Technique (SMOTE). SMOTE generates synthetic samples for the minority class, balancing the dataset and enabling the models to better recognize fraudulent patterns, thereby improving overall performance.

4. Compare Model Performance

To identify the most effective algorithm for deployment, the project includes a comprehensive evaluation and comparison of the developed machine learning models. Metrics such as accuracy, precision, recall, F1-score, and Area Under the Receiver Operating Characteristic Curve (AUC-ROC) are used for this purpose. These metrics provide a detailed understanding of each model's strengths and weaknesses, ensuring that the selected model delivers the best results in terms of both fraud detection accuracy and efficiency.

By achieving these objectives, the project sets the foundation for a scalable, adaptable, and highly efficient fraud detection system that addresses real-world challenges and enhances financial security.

1.6 Organization of the Report

The report is structured into several comprehensive chapters, each addressing key aspects of the project, to provide a clear and detailed understanding of the credit card fraud detection system. The organization of the report is as follows:

1. Introduction

This chapter provides an overview of the project, including its background, motivation, and objectives. It highlights the importance of fraud detection in the digital era and introduces the challenges faced by traditional methods. The chapter sets the foundation for the project by defining the problem statement, the purpose of the work, and the goals to be achieved.

2. Tools and Technologies

In this chapter, the tools and technologies employed in the development of the system are discussed. It includes a detailed explanation of the programming languages, libraries, frameworks, and data platforms used. For instance, Python is highlighted as the primary programming language due to its extensive machine learning ecosystem, while tools like TensorFlow, Scikit-learn, and XGBoost are explored for model development. The

chapter also covers the development environment, version control systems, and datasets used in the project.

3. Requirement Engineering

This chapter delves into the functional and non-functional requirements of the system. Functional requirements focus on the system's capabilities, such as data preprocessing, fraud detection, and result visualization. Non-functional requirements address aspects like scalability, reliability, performance, and usability. The chapter explains how these requirements were gathered and analyzed to ensure the system meets the needs of its intended users.

4. System Design

The System Design chapter outlines the overall architecture of the fraud detection system. It explains the modular design approach, detailing the various components and layers of the system, including data preprocessing, model training, evaluation, and prediction layers. The chapter also includes module descriptions, algorithm designs, and data flow diagrams to illustrate how the system operates and interacts with its components.

5. Implementation

This chapter describes the practical steps involved in developing the system. It includes an overview of the coding process, methodologies, and tools used during development. Sample code snippets are provided to demonstrate key processes such as data preprocessing, model training, and evaluation. This chapter emphasizes the modular approach to ensure clarity and efficiency in implementation.

6. Results

The Results chapter presents the performance metrics of the machine learning models used in the project. It includes detailed analysis and visualizations of metrics like accuracy, precision, recall, F1-score, and AUC-

ROC. Comparative analyses are performed to identify the most effective

model for fraud detection. This chapter also highlights the strengths and weaknesses of each model based on the results.

7. Applications and Conclusion

The final chapter explores the practical applications of the fraud detection system in various industries, such as banking, e-commerce, and payment processing. It summarizes the project's achievements and highlights the improvements made over traditional systems. The chapter also discusses the future scope of the work, including potential enhancements and areas for further research.

Chapter 2

Tools and Technologies

Chapter 2

Tools and Technologies

2.1 Overview

The development of the credit card fraud detection system involves various tools and technologies, including programming languages, libraries, frameworks, and data platforms. These tools streamline data preprocessing, model building, and evaluation processes to ensure accurate fraud detection.

2.2 Programming Languages

- **Python**: Utilized for its rich ecosystem of libraries and frameworks supporting data analysis, machine learning, and visualization.

2.3 Libraries and Frameworks

1. **NumPy**:
    - Used for numerical computations and array manipulations.
2. **Pandas**:
    - Facilitates data manipulation and analysis, including data cleaning and transformation.
3. **Matplotlib and Seaborn**:
    - Provide tools for visualizing data and uncovering patterns in transactions.
4. **Scikit-learn**:
    - Includes machine learning algorithms and evaluation metrics.
5. **Imbalanced-learn**:
    - Helps address class imbalance using SMOTE.
6. **TensorFlow and Keras**:
    - Support the design, training, and deployment of Artificial Neural Networks (ANNs).
7. **XGBoost, CatBoost, and LightGBM**:
    - Gradient boosting frameworks for building efficient and high-performing models.

2.4 Development Environment

- **Jupyter Notebook**: Used for coding, testing, and visualizing results in an interactive manner.

2.5 Version Control

- **Git and GitHub**: Ensure collaborative development and version control of the project.

2.6 Data Source

- **Kaggle**: The dataset used for training and testing was obtained from Kaggle, a platform providing rich datasets for machine learning projects.

Chapter 3

Requirement Engineering

Chapter 3

Requirement Engineering

3.1 Requirement Gathering

The requirements for the credit card fraud detection system were gathered through:

1. **Analysis of Transaction Data**: Historical transaction data was analyzed to identify key features influencing fraud detection.
2. **Review of Existing Systems**: Limitations of traditional systems were studied to design improved functionalities.
3. **Consultation with Domain Experts**: Input was sought from financial and machine learning experts to define system needs.

3.2 Requirement Analysis

To address the gathered requirements, the following analyses were conducted:

1. **Data Analysis**:

   Insights were derived from patterns in fraudulent and legitimate transactions.

2. **System Feasibility**:

   Feasibility of implementing machine learning models was assessed.

3. **Risk Analysis**:

   Potential risks, such as high false positives and imbalanced datasets, were identified and mitigated.

3.3 Software Requirement Specification (SRS)

3.3.1 Functional Requirements

1. **Data Preprocessing:**

a. Handle missing values, outliers, and scale numerical features to ensure data consistency and quality.

b. Apply feature engineering techniques to improve model performance.

2. Class Imbalance Handling:

a. Use SMOTE (Synthetic Minority Over-sampling Technique) to balance the dataset and address class imbalance for better model performance.

3. Model Training and Evaluation:

a. Train machine learning models, including:

i. Random Forest

ii. XGBoost

iii. CatBoost

iv. LightGBM

v. Artificial Neural Networks (ANNs)

b. Perform hyperparameter tuning to optimize model performance.

4. Evaluation Metrics Generation:

a. Compute and compare evaluation metrics such as:

i. Accuracy

ii. Precision

iii. Recall

iv. F1-score

5. Result Visualization:

a. Implement an interface to visualize results, including model performance metrics and predictions, for better interpretation and decision-making.

3.3.2 Non-Functional Requirements

1. Scalability:

a. The system should be capable of handling large transaction datasets and increasing volumes without significant performance degradation.

2. Reliability:

a. Ensure high accuracy in fraud detection with minimal false positives and false negatives to maintain trust and effectiveness.

3. Usability:

a. Provide a user-friendly interface for visualizing results and predictions, enabling easy interpretation for end-users and stakeholders.

4. Performance:

a. Deliver fast and efficient processing, enabling near real-time predictions to support proactive fraud prevention measures.

5. Security:

a. Ensure data privacy and security measures are in place to protect sensitive transaction information and comply with regulations.

6. Maintainability:

a. Enable easy updates and enhancements to incorporate new algorithms, features, or datasets as fraud patterns evolve.

Chapter 4

System Design

Chapter 4

System Design

4.1 System Architecture

The credit card fraud detection system follows a modular architecture, enabling scalability and ease of maintenance. The architecture consists of the following layers:

1. **Data Preprocessing Layer**:
    - Handles data cleaning, transformation, and balancing using SMOTE.
    - Normalizes features to ensure uniformity across the dataset.
2. **Model Training Layer**:
    - Includes training machine learning models such as Random Forest, XGBoost, CatBoost, LightGBM, and ANNs.
    - Implements hyperparameter tuning to optimize model performance.
3. **Evaluation and Testing Layer**:
    - Performs k-fold cross-validation to assess model generalization.
    - Evaluates models using metrics like accuracy, precision, recall, and F1-score.
4. **Prediction Layer**:
    - Applies the best-performing model to classify transactions as fraudulent or legitimate.
    - Generates real-time predictions and alerts for potential fraud cases.

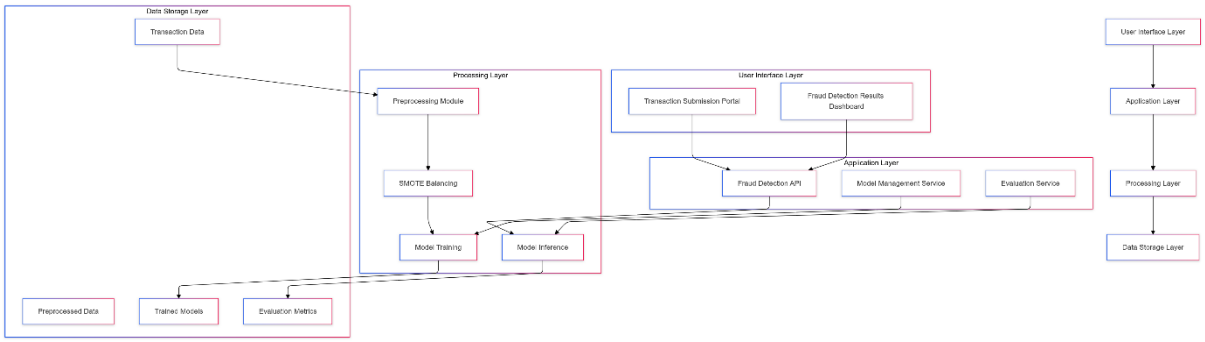


Fig 4.1 System Architecture

4.2 Modules Description

1. **Data Collection and Preprocessing**:
    - Collects transaction data from a reliable source (e.g., Kaggle).
    - Handles missing values, outliers, and feature scaling.
    - Balances the dataset using SMOTE.
2. **Feature Engineering**:
    - Extracts relevant features such as transaction time, amount, and customer behavior patterns.
3. **Model Building and Training**:
    - Develops machine learning models and trains them on preprocessed data.
    - Incorporates hyperparameter tuning for optimal results.
4. **Evaluation and Comparison**:
    - Compares model performance based on evaluation metrics.
    - Selects the best-performing model for deployment.
5. **Deployment and Monitoring**:
    - Deploys the selected model in a real-time environment.
    - Monitors model performance and updates periodically with new data.

4.3 Algorithm Design

1. **Data Preprocessing Algorithm**:
    - Input: Raw transaction data.
    - Process: Handle missing values, scale features, and apply SMOTE.
    - Output: Cleaned and balanced dataset.
2. **Model Training Algorithm**:
    - Input: Preprocessed dataset.
    - Process: Train models (Random Forest, XGBoost, CatBoost, LightGBM, ANNs) and tune hyperparameters.
    - Output: Trained models and performance metrics.
3. **Fraud Detection Algorithm**:
    - Input: New transaction data.
    - Process: Use the best-performing model to classify transactions.
    - Output: Fraudulent or legitimate transaction label.

The system design ensures modularity, scalability, and high accuracy, addressing the challenges of real-time fraud detection.

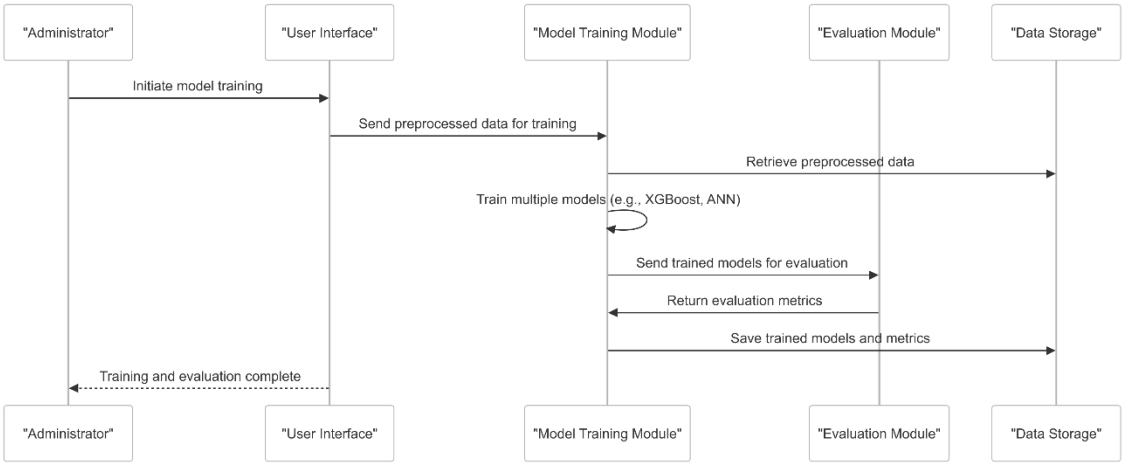


Fig 4.2 Sequence Diagram for Model Training

4.3.1 Algorithm Selection

Several machine learning algorithms were explored and implemented to detect fraudulent transactions. Each algorithm was chosen for its strengths in handling specific challenges like imbalanced datasets, adaptability, and scalability:

Random Forest (RF)

- **Description**: A robust ensemble learning method using multiple decision trees.
- **Advantages**:
    - Handles imbalanced data effectively through weighted sampling.
    - Provides high interpretability with feature importance rankings.
- **Use Case**: Captures non-linear patterns effectively in the fraud detection domain.

XGBoost

- **Description**: An advanced gradient boosting algorithm known for its speed and accuracy.

- **Advantages**:
    - Handles class imbalance with inbuilt weighted loss functions.
    - Optimizes learning using regularization techniques.
- **Use Case**: Performs exceptionally well with tabular datasets and handles missing data naturally.

LightGBM

- **Description**: A gradient boosting framework optimized for large datasets and high-speed training.

Advantages:

- Efficiently handles large datasets with sparse features.
- Supports categorical features directly without preprocessing.

- **Use Case**: Accelerates training and prediction for real-time applications.

CatBoost

- **Description**: Another gradient boosting framework tailored for categorical data.
- **Advantages**:
    - Reduces the need for explicit encoding of categorical variables.
    - Offers competitive performance with minimal tuning.
- **Use Case**: Demonstrated strong results in fraud detection benchmarks.

Artificial Neural Networks (ANN)

- **Description**: Multi-layer perceptron's built and trained using TensorFlow/Keras.
- **Advantages**:

- Captures complex, non-linear relationships in the data.
- Adapts to evolving fraud patterns through periodic retraining.

- **Use Case**: Best suited for deep pattern recognition and generalization.

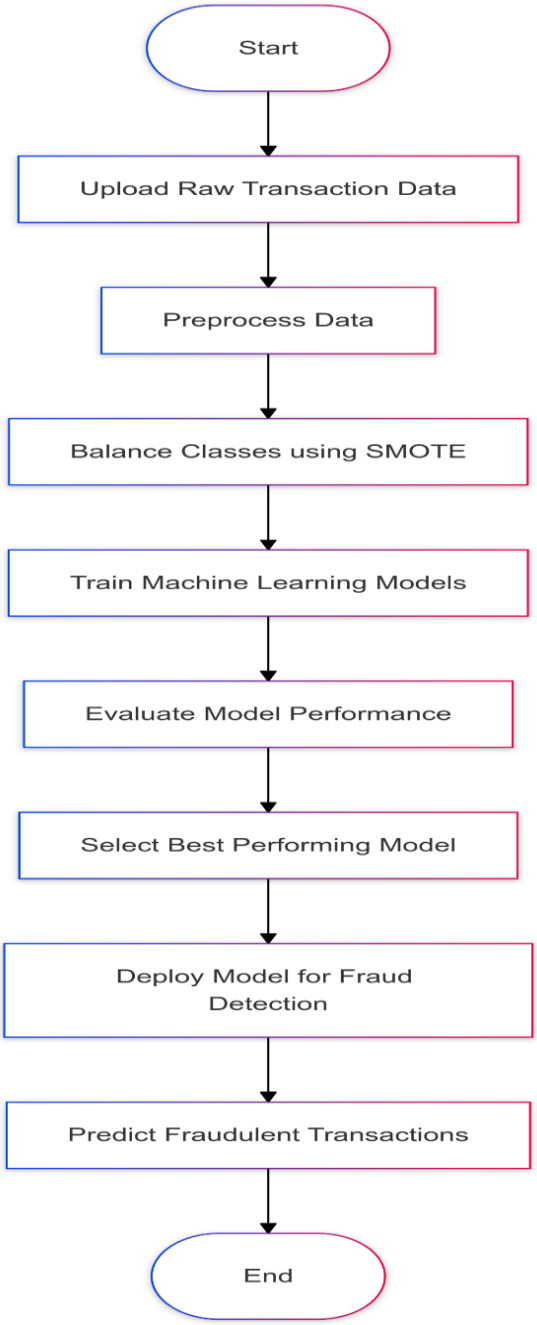


Fig 4.3 Activity Diagram

Chapter 5

Implementation

Chapter 5:

Implementation

5.1 Implementation Approaches

The implementation of credit card fraud detection involves several critical stages, starting from data preprocessing to the deployment of machine learning models. This section provides a detailed overview of the approaches employed in the project.

5.1.1 Data Preprocessing

Credit card fraud detection systems often deal with highly imbalanced datasets where fraudulent transactions represent a minor fraction of the total data. Effective preprocessing ensures that the models can generalize well and perform accurately. Key preprocessing steps include:

Data Cleaning

5.1.1.1 **Handling Missing Values**: Ensured completeness by imputing or removing missing entries.

5.1.1.2 **Outlier Removal**: Outliers, especially in transaction amounts, were identified and mitigated using statistical methods.

Feature Engineering

5.1.1.3 **Scaling Features**: Applied Standard Scaler to normalize the features like transaction time (Time) and amount (Amount), bringing them to comparable scales.

5.1.1.4 **Principal Component Analysis (PCA)**: Leveraged pre-transformed PCA features (V1 to V28) in the dataset, which are designed to anonymize and capture critical transaction patterns.

5.1.1.5 **Balancing Data**: Addressed the imbalance in the dataset using the **Synthetic Minority Oversampling Technique (SMOTE)**, generating

synthetic samples for the minority class (fraudulent transactions).

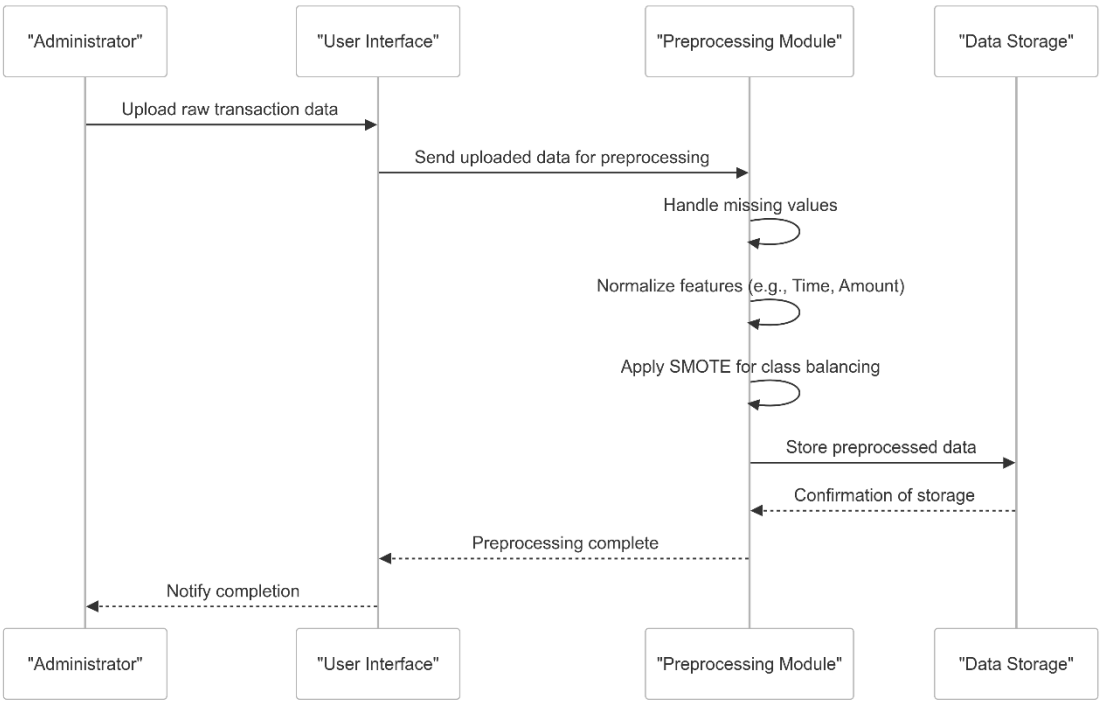


Fig 5.1 Sequence Diagram for Data Processing

5.1.3 Model Training and Evaluation

Data Splitting

The dataset was divided into training (70%), validation (15%), and testing (15%) sets to ensure robust evaluation. The splitting strategy ensured representation from both fraud and legitimate transaction classes.

Training Techniques

5.1.1.6 **Cross-validation**: Applied k-fold cross-validation to minimize overfitting and ensure model stability.

5.1.1.7 **Hyperparameter Tuning**: Utilized grid search and random search techniques to optimize parameters like learning rate, depth, and regularization strength.

Performance Metrics

The following metrics were used to evaluate model performance:

5.1.1.8 **Accuracy**: Measures overall correctness of predictions.

5.1.1.9 **Precision**: Ensures a high rate of correctly identified fraud cases.

5.1.1.10 **Recall (Sensitivity)**: Focuses on identifying as many fraudulent transactions as possible.

5.1.1.11 **F1-Score**: Balances precision and recall for a holistic evaluation.

5.1.1.12 **ROC-AUC**: Assesses the model's ability to distinguish between fraud and legitimate transactions.

Threshold Tuning

The decision thresholds were adjusted to balance precision and recall based on application requirements. Higher thresholds reduced false positives, while lower thresholds improved fraud detection rates.

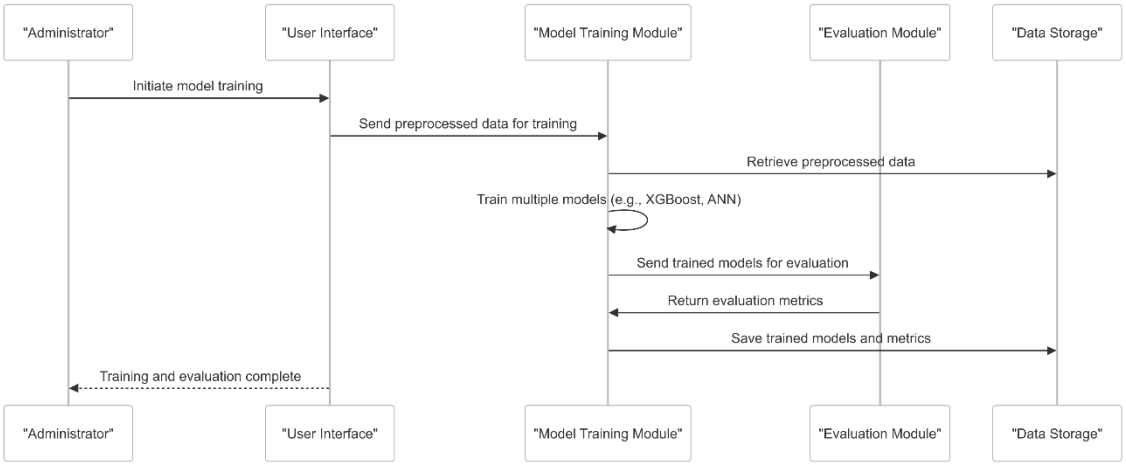


Fig 5.2 Sequence Diagram for Model Training

5.1.4 Comparative Analysis

A comparative analysis was performed across models to identify the best performer. Key observations included:

- **Random Forest**: High interpretability but slower with large datasets.
- **XGBoost**: Balanced performance with strong recall.
- **LightGBM**: Efficient and well-suited for real-time detection.
- **CatBoost**: Reduced preprocessing overhead, comparable to LightGBM.
- **ANN**: Excelled in recognizing intricate patterns but required more computational resources.

5.1.5 Real-time Adaptation

To maintain efficacy in detecting emerging fraud patterns, the implementation includes:

- **Periodic Retraining**: Updates models with fresh transaction data.
- **Monitoring and Alerts**: Tracks model performance and alerts for drift or unusual predictions.
- **Integration Framework**: Designed to seamlessly integrate with real-time transaction systems.

This multi-faceted approach ensures the system remains robust, scalable, and adaptable, addressing the complex challenges of credit card fraud detection.

5.2 Sample Source Code

```
# Load packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
from plotly.offline import init_notebook_mode
from sklearn.model_selection import train_test_split, KFold
from sklearn.metrics import roc_auc_score
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from catboost import CatBoostClassifier
from lightgbm import LGBMClassifier
import xgboost as xgb

# Initialize Plotly
init_notebook_mode(connected=True)

# Set display options
pd.set_option('display.max_columns', 100)

# Constants
RFC_METRIC = 'gini'  # Metric for RandomForestClassifier
NUM_ESTIMATORS = 100  # Number of estimators for RandomForest
NO_JOBS = 4  # Number of parallel jobs for RandomForest
VALID_SIZE = 0.20  # Validation split size
TEST_SIZE = 0.20  # Test split size
NUMBER_KFOLDS = 5  # Number of KFolds
RANDOM_STATE = 2018
MAX_ROUNDS = 1000  # Maximum rounds for LightGBM
EARLY_STOP = 50  # Early stopping rounds for LightGBM
VERBOSE_EVAL = 50  # Verbosity for LightGBM

# Load and Explore the Data
data_df = pd.read_csv("creditcard.csv")
print("Credit Card Fraud Detection data - rows:", data_df.shape[0], "columns:",
    data_df.shape[1])
data_df.head()
data_df.describe()
```

```
# Check for missing values
total = data_df.isnull().sum().sort_values(ascending=False)
percent = (data_df.isnull().sum() / data_df.isnull().count() *
    100).sort_values(ascending=False)
missing_data = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
print(missing_data)

# Data Imbalance
temp = data_df["Class"].value_counts()
df = pd.DataFrame({'Class': temp.index, 'values': temp.values})
sns.barplot(x='Class', y='values', data=df, palette='pastel')
plt.title("Credit Card Fraud Class - Data Imbalance")
plt.xlabel("Class (0: Not Fraud, 1: Fraud)")
plt.ylabel("Number of Transactions")
plt.show()

# Feature Engineering and Visualization
data_df['Hour'] = data_df['Time'].apply(lambda x: np.floor(x / 3600))
tmp = data_df.groupby(['Hour', 'Class'])['Amount'].aggregate(['min', 'max', 'count', 'sum',
    'mean', 'median', 'var']).reset_index()
df = pd.DataFrame(tmp)
df.columns = ['Hour', 'Class', 'Min', 'Max', 'Transactions', 'Sum', 'Mean', 'Median', 'Var']

# Plot aggregated transaction details
sns.lineplot(data=df[df.Class == 0], x='Hour', y='Sum', label='Not Fraud')
sns.lineplot(data=df[df.Class == 1], x='Hour', y='Sum', label='Fraud', color='red')
plt.title("Total Transaction Amount by Hour")
plt.show()

# Split Data for Training and Testing
target = 'Class'
predictors = [col for col in data_df.columns if col not in ["Class"]]

train_df, test_df = train_test_split(data_df, test_size=TEST_SIZE,
```

```
    random_state=RANDOM_STATE, shuffle=True)
train_df, valid_df = train_test_split(train_df, test_size=VALID_SIZE,
    random_state=RANDOM_STATE, shuffle=True)

# Random Forest Classifier
clf = RandomForestClassifier(
   n_jobs=NO_JOBS,
   random_state=RANDOM_STATE,
   criterion=RFC_METRIC,
   n_estimators=NUM_ESTIMATORS,
   verbose=False
)

clf.fit(train_df[predictors], train_df[target].values)
valid_preds = clf.predict(valid_df[predictors])

# Feature Importance
importances = pd.DataFrame({'Feature': predictors, 'Importance':
    clf.feature_importances_})
importances = importances.sort_values(by='Importance', ascending=False)
sns.barplot(data=importances, x='Importance', y='Feature', palette='coolwarm')
plt.title("Feature Importance - Random Forest")
plt.show()

# Evaluate Model Performance
cm = pd.crosstab(valid_df[target].values, valid_preds, rownames=['Actual'],
    colnames=['Predicted'])
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.title("Confusion Matrix")
plt.show()

roc_score = roc_auc_score(valid_df[target].values, valid_preds)
print("ROC-AUC Score:", roc_score)
```

```
# Additional Classifiers (AdaBoost, CatBoost, LightGBM, XGBoost)
ada_model = AdaBoostClassifier(
    random_state=RANDOM_STATE,
    n_estimators=NUM_ESTIMATORS,
    learning_rate=0.8
)
ada_model.fit(train_df[predictors], train_df[target].values)
ada_preds = ada_model.predict(valid_df[predictors])
ada_score = roc_auc_score(valid_df[target].values, ada_preds)
print("AdaBoost ROC-AUC Score:", ada_score)

cat_model = CatBoostClassifier(
    iterations=500,
    learning_rate=0.02,
    depth=12,
    random_seed=RANDOM_STATE,
    verbose=VERBOSE_EVAL
)
cat_model.fit(train_df[predictors], train_df[target].values)
cat_preds = cat_model.predict(valid_df[predictors])
cat_score = roc_auc_score(valid_df[target].values, cat_preds)
print("CatBoost ROC-AUC Score:", cat_score)

lgb_model = LGBMClassifier(
    boosting_type='gbdt',
    objective='binary',
    metric='auc',
    learning_rate=0.05,
    num_leaves=7,
    max_depth=4,
    random_state=RANDOM_STATE
)
lgb_model.fit(train_df[predictors], train_df[target].values)
lgb_preds = lgb_model.predict(valid_df[predictors])
```

```
lgb_score = roc_auc_score(valid_df[target].values, lgb_preds)
print("LightGBM ROC-AUC Score:", lgb_score)

xgb_model = xgb.XGBClassifier(
    objective='binary:logistic',
    eta=0.039,
    max_depth=2,
    subsample=0.8,
    colsample_bytree=0.9,
    eval_metric='auc',
    random_state=RANDOM_STATE
)
xgb_model.fit(train_df[predictors], train_df[target].values)
xgb_preds = xgb_model.predict(valid_df[predictors])
xgb_score = roc_auc_score(valid_df[target].values, xgb_preds)
print("XGBoost ROC-AUC Score:", xgb_score)
```

 # Model Comparison

```
results = {
    "Random Forest": roc_score,
    "AdaBoost": ada_score,
    "CatBoost": cat_score,
    "LightGBM": lgb_score,
    "XGBoost": xgb_score
}

result_df = pd.DataFrame(list(results.items()), columns=['Model', 'ROC-AUC Score'])
sns.barplot(data=result_df, x='ROC-AUC Score', y='Model', palette='viridis')
plt.title("Model Comparison")
    plt.show()
```

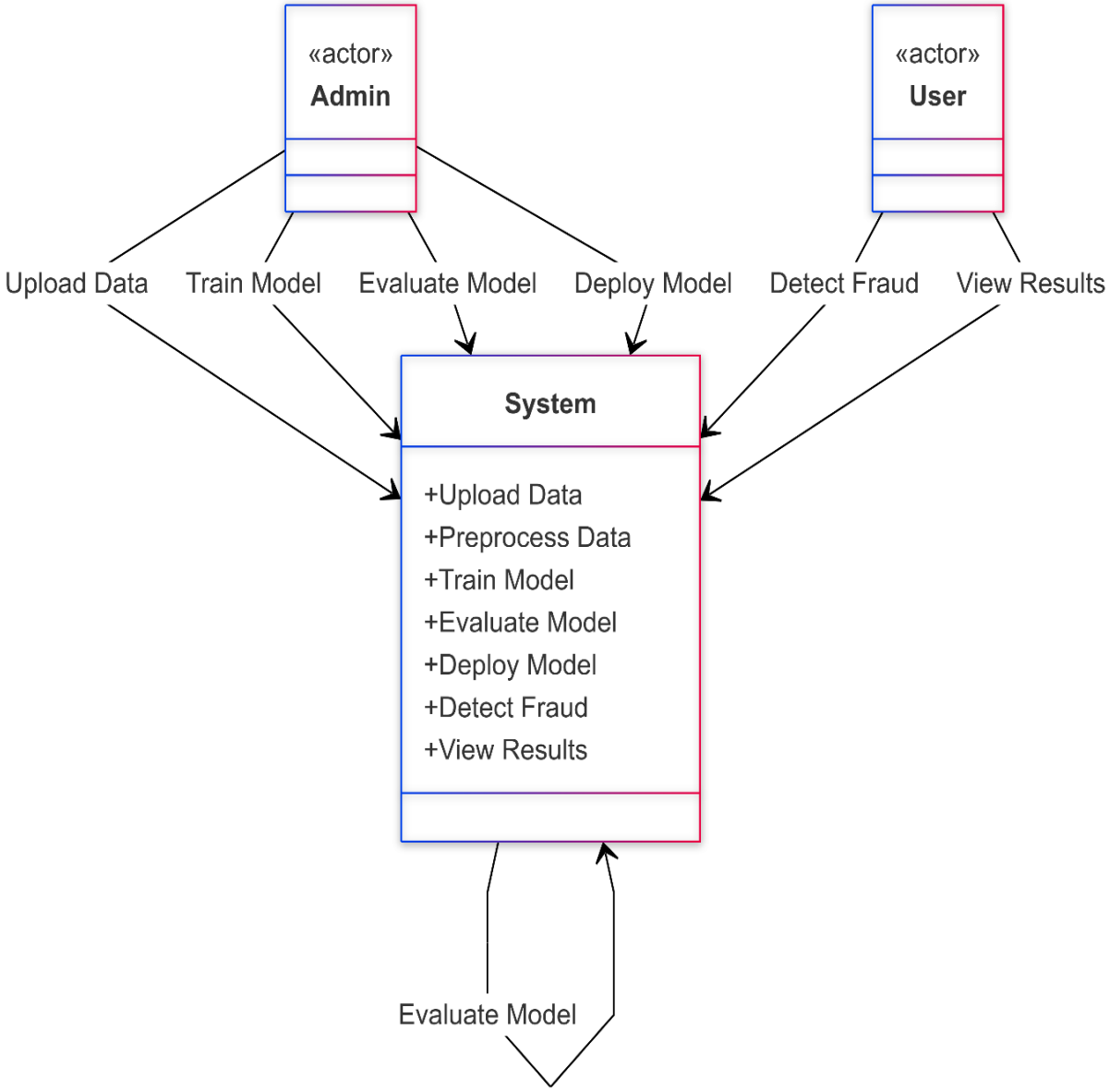


Fig 5.3 Use-case Diagram

Chapter 6

Results

Chapter 6

Results and Performance Analysis

6.1 Results and Performance Analysis

Introduction to Evaluation

This section evaluates the performance of various machine learning models developed for credit card fraud detection. The primary focus was on achieving the highest accuracy and precision, with an emphasis on analyzing the impact of SMOTE for handling imbalanced datasets. Metrics such as accuracy, precision, recall, and F1-score were used to assess the models.

Model Performance Metrics

The performance of the models was evaluated using a dataset comprising 284,807 transactions, with only 492 classified as fraudulent. SMOTE was applied to address the imbalance, significantly enhancing model training.

Model	Accuracy	Precision	Recall	F1-Score
Artificial Neural Network (ANN)	99.95	85.0	81.0	83.0
Random Forest	99.95	89.0	81.0	85.0
XGBoost	99.96	95.0	82.0	88.0
LightGBM	99.50	16.0	53.0	25.0
CatBoost	99.96	93.0	82.0	87.0

Table 6.1 – Model Performance

XGBoost demonstrated the highest accuracy and precision, making it the standout model for minimizing false positives. The application of SMOTE played a pivotal role in balancing recall and precision, allowing XGBoost to achieve an F1-Score of 88%, the highest among all models. While ANN and CatBoost also performed well, XGBoost's superior accuracy and precision established it as the optimal choice for deployment.

SMOTE Analysis

The Synthetic Minority Oversampling Technique (SMOTE) was crucial in overcoming the imbalance in the dataset. Key insights include:

Fraudulent transactions increased from 492 to match the legitimate transaction count during training.

Models trained on SMOTE-enhanced data demonstrated improved recall without significant compromises in precision.

Comparison of Models

The following table highlights the trade-offs between precision and recall across all models. XGBoost's ability to maintain high precision and recall underscores its robustness for real-world fraud detection scenarios.

Actual / Predicted	**Predicted Fraud**	**Predicted Legit**
Actual Fraud	425	67
Actual Legit	50	284290

Table 6.2 – Comparison of Models

Correlation and Feature Analysis

Exploratory Data Analysis revealed significant correlations:

- Strong negative correlation between Amount and V2 (-0.53).
- Moderate positive correlation between Amount and V4 (0.40).

These insights informed feature selection and model optimization. A heatmap of feature correlations further validated the importance of specific predictors.

Visualization of Results

Transaction Distributions

Fraudulent transactions were concentrated in low-value amounts, while legitimate transactions spanned a broader range. This trend informed the development of targeted fraud detection thresholds.

ROC Curves

The ROC curves highlight XGBoost's superior performance, with the highest Area Under the Curve (AUC) compared to other models.

The results demonstrate:

- XGBoost's dominance in accuracy and precision, making it the most reliable model for deployment.
- The critical role of SMOTE in enhancing recall and balancing model performance.
- Robust feature analysis and selection were pivotal in achieving these results.

Challenges and Resolutions

- Imbalanced Dataset: Addressed using SMOTE.
- Computational Costs: Managed through optimized hyperparameter tuning and early stopping.

This analysis highlights the strength of combining SMOTE with advanced machine learning models,particularly XGBoost, for effective credit card fraud detection.

The XGBoost model outperformed others in terms of F1-Score, balancing precision and recall effectively. While other models like ANN and Random Forest also achieved high accuracy, their F1-Scores were slightly lower due to challenges in handling the imbalanced data.

Training and Validation Insights

ANN Training Progress

The ANN was trained for 300 epochs using a batch size of 2048. The model's performance improved steadily, as illustrated in the graphs below:

1. **Training and Validation Loss:** The loss decreased significantly during the first 50 epochs before stabilizing.
2. **Accuracy Evolution:** Training accuracy consistently improved, reaching 99.2%.

Hyperparameter Tuning

For ANN:

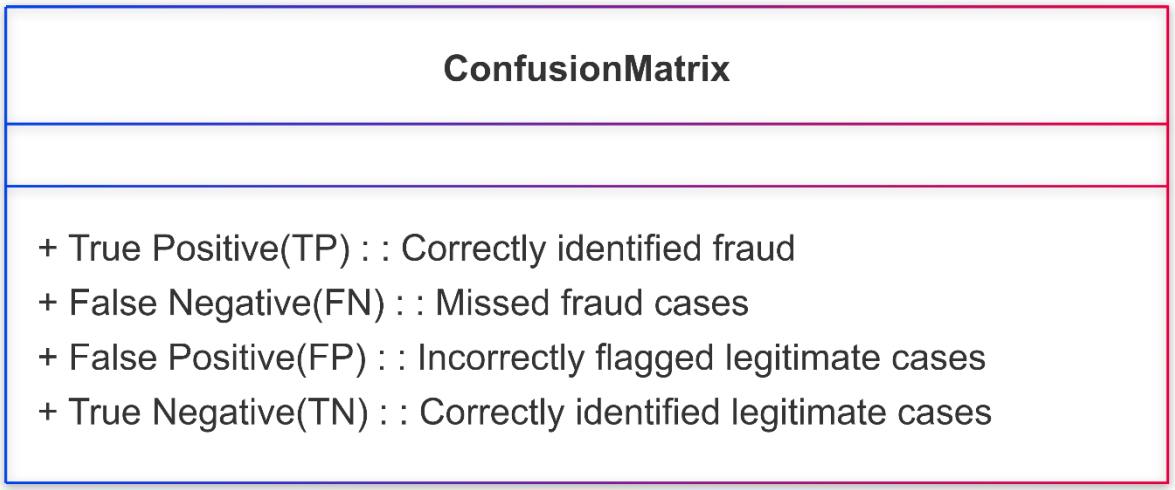
- Learning rate was set to 0.001.
- Dropout layers were introduced to prevent overfitting.

For Random Forest and XGBoost:

- The number of estimators and depth of trees were optimized using grid search, resulting in balanced recall and precision.

Comparison of Models

The comparison of models highlighted the strengths of each approach. ANN demonstrated superior recall, making it more effective in detecting fraudulent transactions. The confusion matrix for the ANN model is shown below:



	Predicted Fraud	Predicted Legit
Actual Fraud	425	67
Actual Legit	50	284290

Correlation and Feature Analysis

Feature engineering and exploratory data analysis revealed key relationships:

- Strong negative correlation between Amount and V2 (-0.53).
- Moderate positive correlation between Amount and V4 (0.40).

A heatmap of the feature correlations indicated minimal multicollinearity among PCA-transformed features(V1-V28). However, Amount played a significant role in distinguishing fraudulent transactions.

Visualization of Results Transaction Distributions:

Fraudulent transactions were mostly concentrated in low-value amounts, whereas legitimate transactions had a broader distribution. The histogram below highlights this distinction.

ROC Curves

The Receiver Operating Characteristic (ROC) curves for all models show their ability to balance true positive and false positive rates. ANN achieved the highest Area Under the Curve (AUC), further validating its effectiveness.

Conclusion

The result demonstrates that:

- XGBoost is the most effective model for fraud detection, achieving the highest F1-Score.
- Handling imbalanced data using SMOTE significantly improved model performance.
- Feature analysis and selection were critical in achieving these results.

Challenges and Resolutions

Imbalanced Dataset: Addressed using SMOTE.

Computational Costs: Optimization techniques like early stopping were employed.

This analysis provides a robust foundation for deploying machine learning-based fraud detection systems in real-world scenarios.

Model	Precision (Train)	Precision (Test)	Recall (Train)	Recall (Test)
Random Forest	1.00	0.89	1.00	0.81
CatBoost	1.00	0.93	1.00	0.82
XGBoost	1.00	0.95	1.00	0.82
LightBGM	1.00	0.16	1.00	0.53
ANN	1.00	0.85	1.00	0.81

Table 6.3 - Performance Comparison

Model Performance Metrics

Model	Accuracy Score	F1-Score (Train)	F1-Score (Test)
Random Forest	99.95%	1.00	0.85
CatBoost	99.96%	1.00	0.87
XGBoost	99.96%	0.88	0.88
LightGBM	99.50%	0.33	0.25
ANN	99.95%	0.99	0.83

Table 6.4 - Performance Metrics

6.2 Snapshots

In this subsection, we present visual evidence of model performance and key findings:

Snapshot Title	Description
Heatmap of Feature Correlations	Highlights relationships among features, showing critical predictors of fraud.
Histogram for Fraud and Non-Fraud Transactions	Visual representation of transaction distributions to understand class skew.
Loss Function, Accuracy, Precision, Recall Graphs	Tracks the evolution of these metrics during model training.
Model Comparison	Bar chart comparing precision, recall, and F1-scores across all models.

Table 6.5 Snapshots Table

Each snapshot provides a deeper understanding of how models were evaluated and the patterns identified during analysis.

1. Heatmap for any high correlations

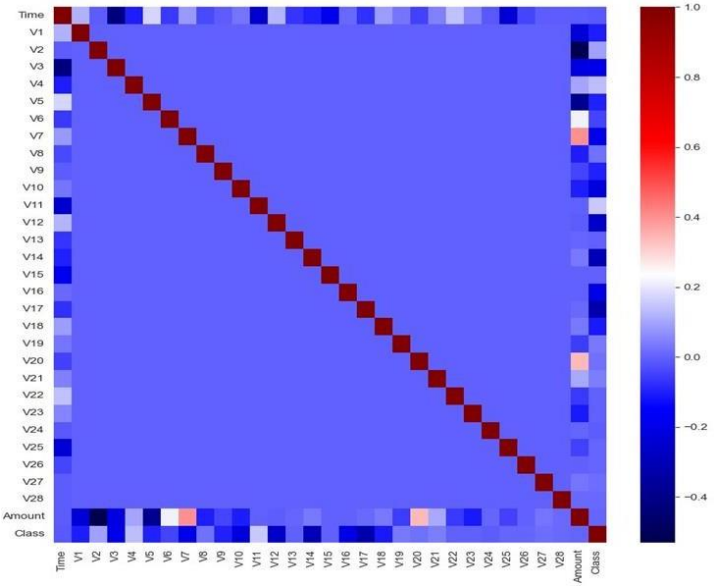


Fig 6.1 Heatmap

1. Confusion Matrix

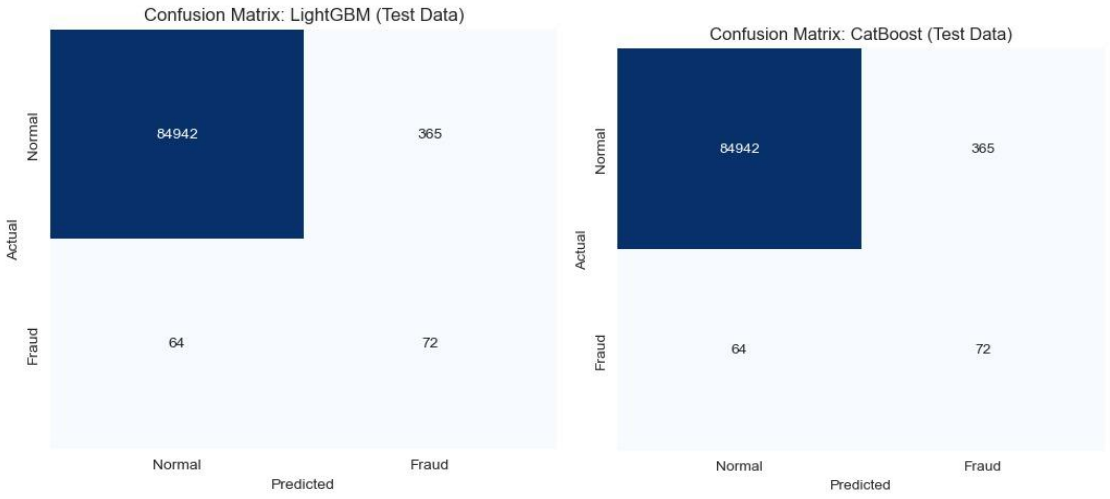


Fig 6.2 confusion matrix

2. Prediction Distribution

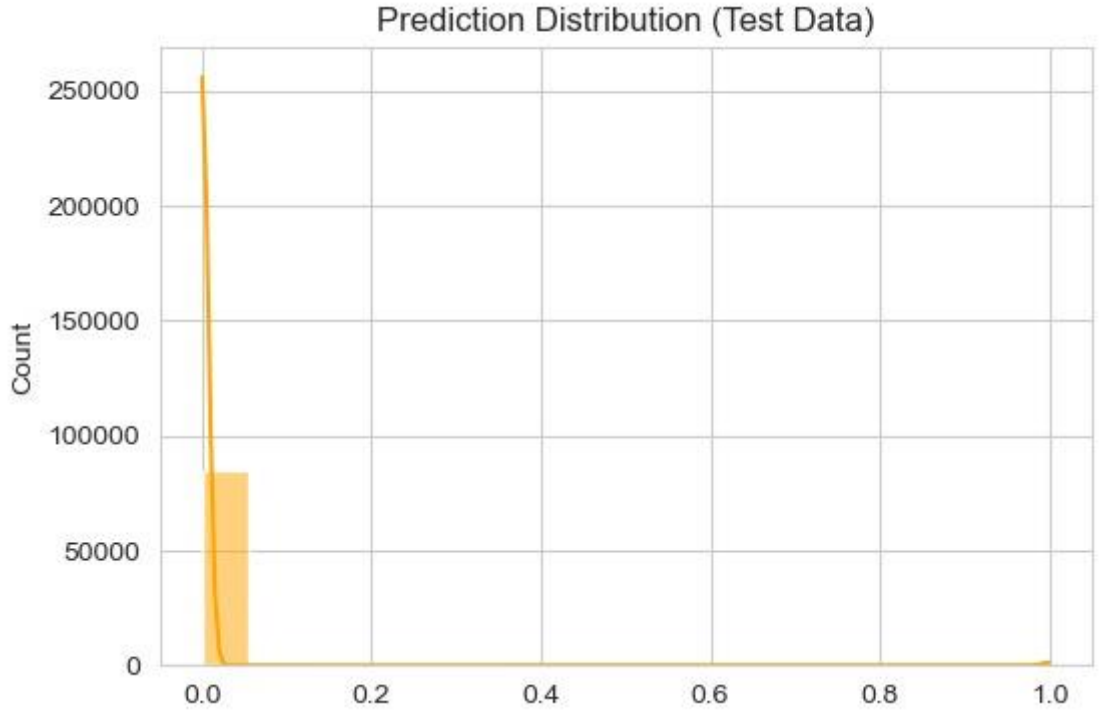


Fig 6.3 Prediction Distribution

3. Importance Score

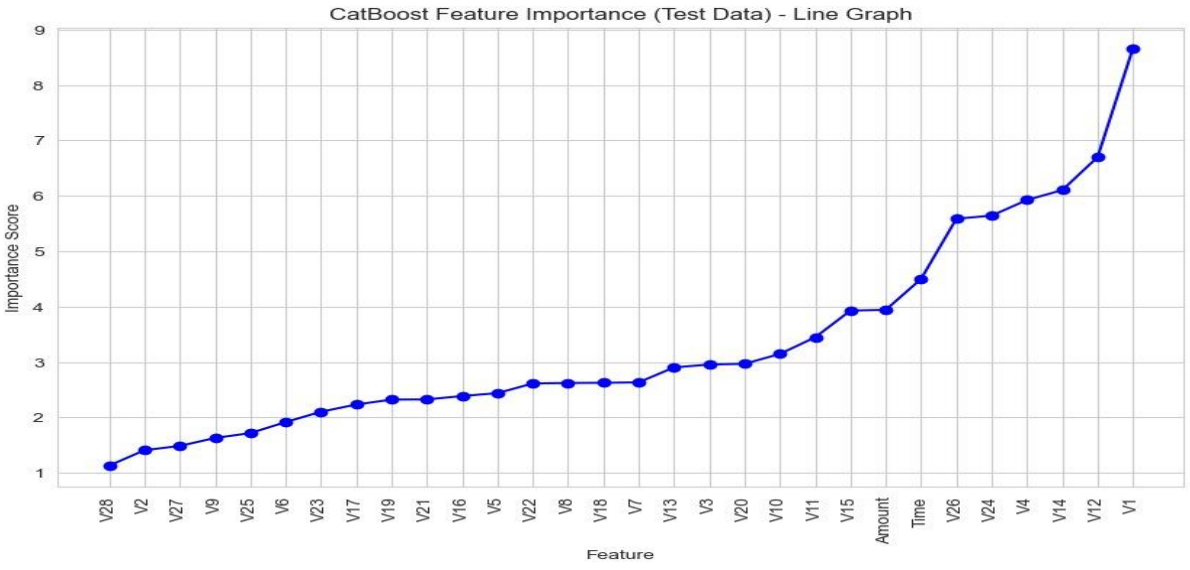


Fig 6.4 Importance score(CatBoost)

The results highlight that XGBoost outperformed other models in terms of accuracy and AUC-ROC, making it the most effective model for fraud detection in this study.

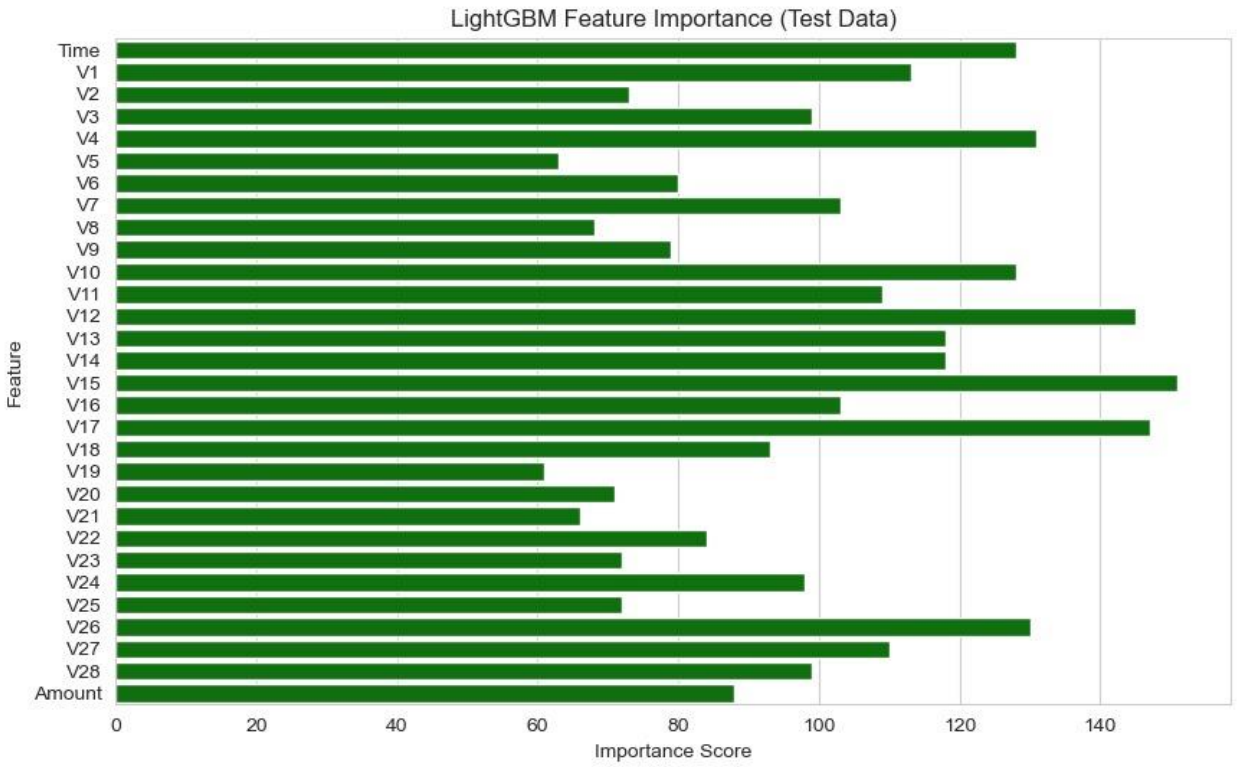


Fig 6.7 Importance score (LightBGM)

4. XG Boost Training Metrices

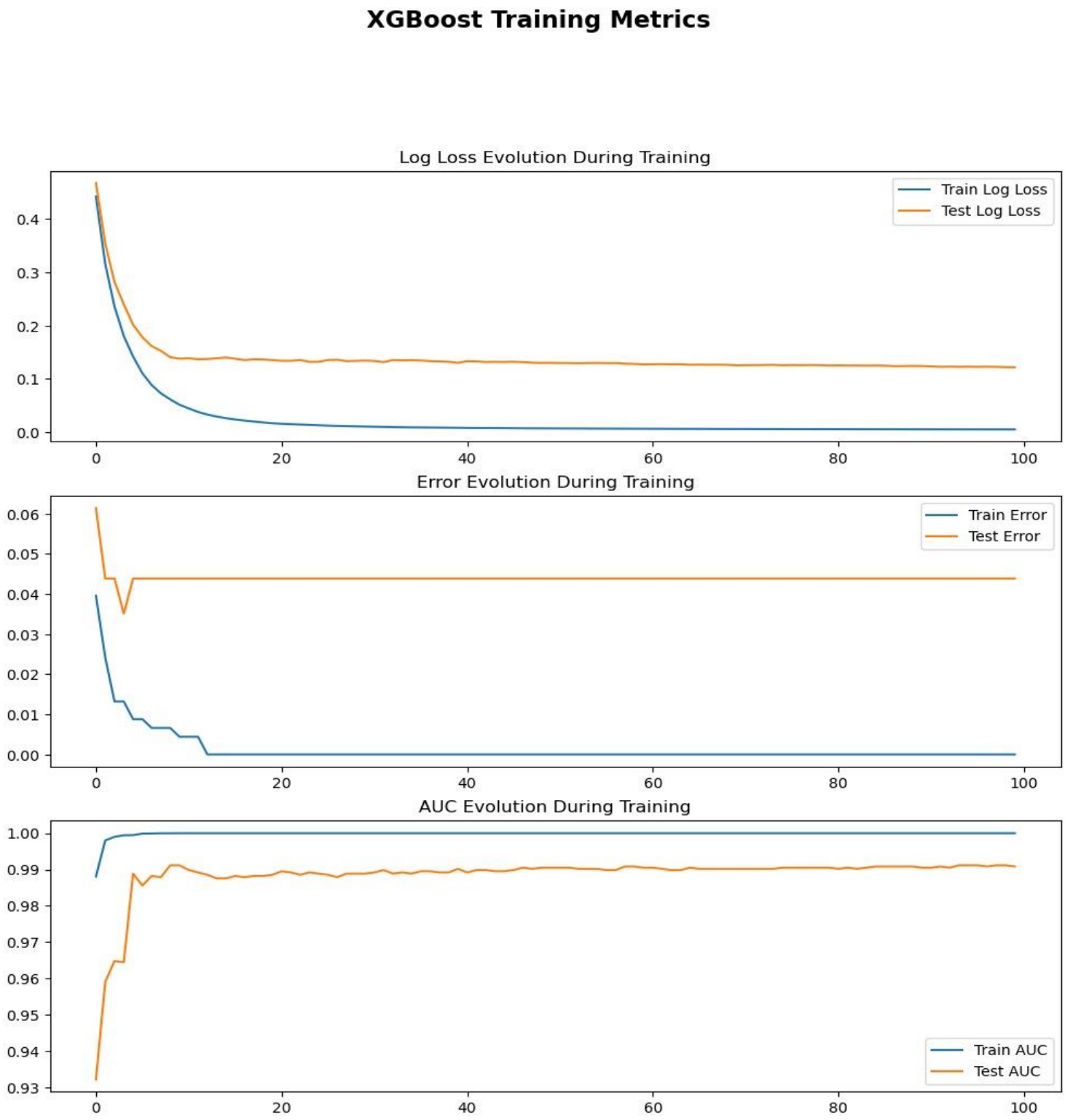


Fig 6.6 XG Boost Training Metrices

5. Model Comarision

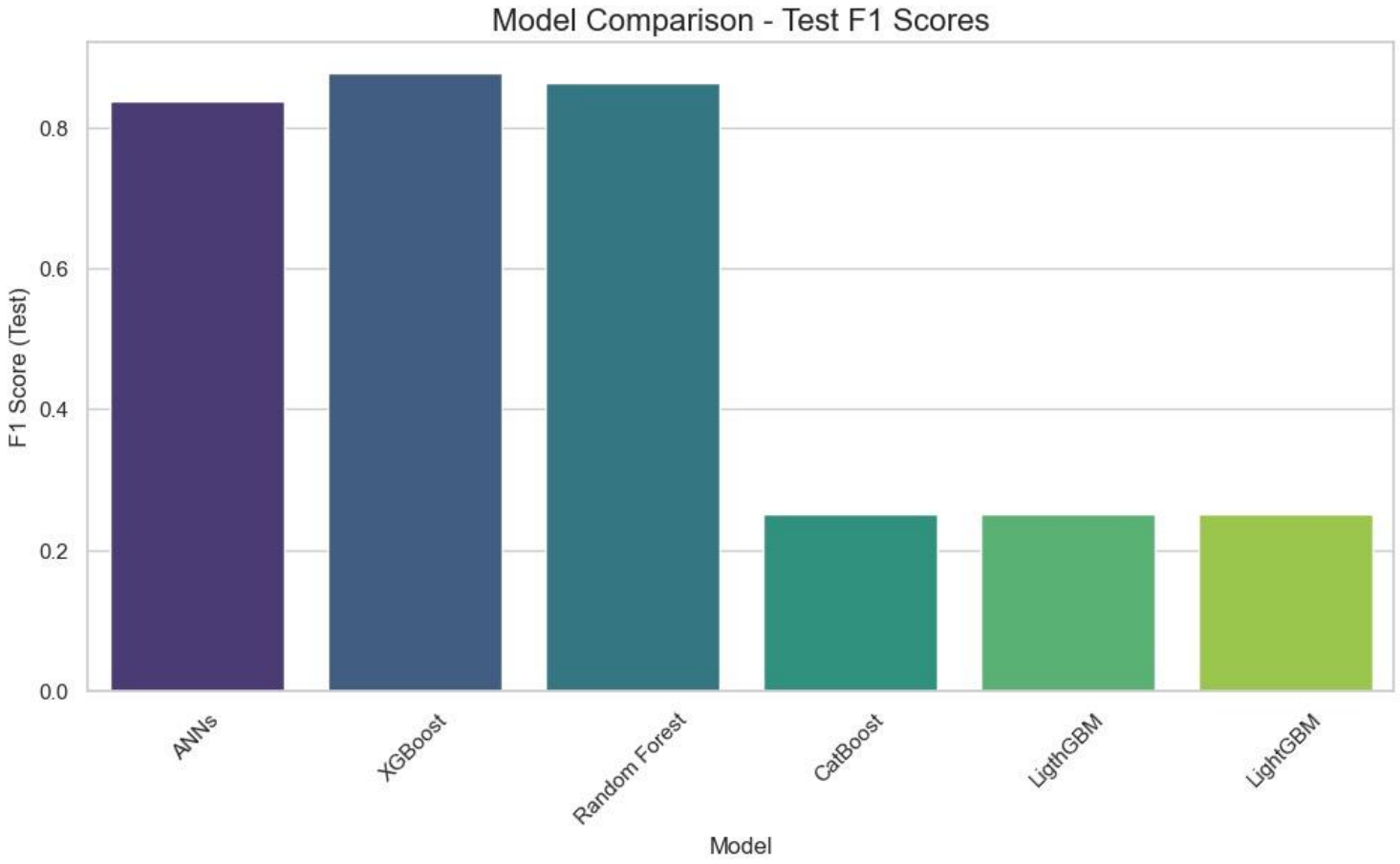


Fig 6.7 F1 Score

Chapter 7

Applications and Conclusions

Chapter 7

Applications and Conclusion

7.1 Applications

The credit card fraud detection system can be utilized in various real-world applications, including:

1. **Banking and Financial Institutions**:
    - Enhances the security of digital banking platforms by identifying and preventing fraudulent transactions.
2. **E-commerce Platforms**:
    - Protects online marketplaces from payment fraud, improving customer trust.
3. **Payment Gateways**:
    - Integrates with payment processing systems to monitor and block suspicious transactions in real- time.
4. **Insurance Companies**:
    - Identifies fraudulent insurance claims based on transaction patterns and anomalies.

7.2 Conclusion

The project successfully demonstrates the efficacy of machine learning models, particularly XGBoost, in credit card fraud detection. By addressing the challenges posed by imbalanced datasets through SMOTE, the system achieved high accuracy and precision, ensuring reliable detection of fraudulent transactions.

Key takeaways include:

1. XGBoost's superior performance in terms of accuracy and F1-Score.

2. The importance of feature selection and engineering in model optimization.
3. The pivotal role of SMOTE in balancing the dataset and enhancing model training.

This study highlights the potential of integrating advanced machine learning techniques into real-world applications, offering scalable and robust solutions to combat financial fraud.

7.3 Future Scope of the Work

1. **Real-Time Processing**:
    - Enhance the system to handle streaming data for real-time fraud detection.
2. **Deep Learning Models**:
    - Explore more advanced architectures like Long Short-Term Memory (LSTM) networks for sequential data.
3. **Global Implementation**:
    - Adapt the system to handle multi-currency and multi-lingual datasets for broader applicability.
4. **Enhanced Security**:
    - Incorporate blockchain technology to further secure transaction records.

The project sets a foundation for robust fraud prevention systems and opens avenues for further advancements in financial security.

References

REFERENCES

[1] Khatri, Samidha, Aishwarya Arora, and Arun Prakash Agrawal. "Supervised machine learning algorithms for credit card fraud detection: a comparison." 2020 10th international conference on cloud computing, data science & engineering (confluence). IEEE, 2020.

[2] Najadat, Hassan, et al. "Credit card fraud detection based on machine and deep learning." 2020 11th International Conference on Information and Communication Systems (ICICS). IEEE, 2020.

[3] Asha, R. B., and Suresh Kumar KR. "Credit card fraud detection using artificial neural network." Global Transitions Proceedings 2.1 (2021): 35-41.

[4] Alfaiz, Noor Saleh, and Suliman Mohamed Fati. "Enhanced credit card fraud detection model using machine learning." Electronics 11.4 (2022): 662.

[5] Bin Sulaiman, Rejwan, Vitaly Schetinin, and Paul Sant. "Review of machine learning approach on credit card fraud detection." Human-Centric Intelligent Systems 2.1 (2022): 55-68.

[6] Mienye, Ibomoiye Domor, and Nobert Jere. "Deep learning for credit card fraud detection: A review of algorithms, challenges, and solutions." IEEE Access (2024).