A PROJECT REPORT ON

Credit Card Fraud Detection Using Machine Learning

Submitted in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

In

Computer Science and Engineering

By

K.Vyshnavi	**(21A81A05M8)**
K.L.S.Abhinaya	**(21A81A05M9)**
K.H.Lakshmi	**(22A85A0521)**
P.Lithish Kumar	**(22A85A0524)**
S.Bharath Kumar	**(21A81A05P7)**

Under the Esteemed Supervision of

Mrs. P.Ujwala Sai M.Tech

Assistant Professor

Department of Computer Science and Engineering (Accredited by N.B.A.)
SRI VASAVI ENGINEERING COLLEGE(Autonomous) (Affiliated to JNTUK, Kakinada) Pedatadepalli, Tadepalligudem-534101, A.P 2023-24
SRI VASAVI ENGINEERING COLLEGE (Autonomous)

Department Of Computer Science and Engineering

Pedatadepalli, Tadepalligudem

Certificate

This is to certify that the Project Report entitled "**Credit Card Fraud Detection Using Machine Learning**" submitted by **K.Vyshnavi (21A81A05M8), K.L.S.Abhinaya (21A81A05M9) , K. Harshitha (22A85A0521) , P.Lithish Kumar (22A85A0524), S.Bharath Kumar (21A81A05P7)** for the award of the degree of Bachelor of Technology in the Department of Computer Science and Engineering during the academic year 2023-2024.

Name of Project Guide

Mrs. P.Ujwala Sai M.Tech
Assistant Professor

Head of the Department

Dr. D. Jaya Kumari M.Tech.,Ph.D.
Professor & HOD.

External Examiner

DECLARATION

We hereby declare that the project report entitled "**Credit Card Fraud Detection Using Machine Learning**" submitted by us to Sri Vasavi Engineering College (Autonomous), Tadepalligudem, affiliated to JNTUK Kakinada in partial fulfilment of the requirement for the award of the degree of B.Tech in Computer Science and Engineering is a record of Bonafide project work carried out by us under the guidance of **Mrs. P.Ujwala Sai M.Tech,** Assistant Professor. We further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree in this institute or any other institute or University.

Project Associate

K.Vyshnavi (21A81A05M8)

K.L.S.Abhinaya (21A81A05M9)

K.Harshitha (22A85A0521)

P.Lithish Kumar (22A85A0524)

S.Bharath Kumar (21A81A05P7)

ACKNOWLEDGEMENT

First and foremost, we sincerely salute to our esteemed institute **SRI VASAVI ENGINEERING COLLEGE,** for giving us this golden opportunity to fulfill our warm dream to become an engineer.

Our sincere gratitude to our project guide **Mrs. P.Ujwala Sai M.Tech,** Assistant Professor, Department of Computer Science and Engineering**,** for her timely cooperation and valuable suggestions while carrying out this project.

We express our sincere thanks and heartful gratitude to **Dr. D. Jaya Kumari**, Professor & Head of the Department of Computer Science and Engineering, for permitting us to do our project.

We express our sincere thanks and heartful gratitude to **Dr. G.V.N.S.R. Ratnakara Rao**, Principal, for providing a favourable environment and supporting us during the development of this project.

Our special thanks to the management and all the teaching and non-teaching staff members, Department of Computer Science and Engineering, for their support and cooperation in various ways during our project work. It is our pleasure to acknowledge the help of all those respected individuals.

We would like to express our gratitude to our parents, friends who helped to complete this project.

Project Associate

K.Vyshnavi	(21A81A05M8)
K.L.S.Abhinaya	(21A81A05M9)
K.Harshitha	(22A85A0521)
P.Lithish Kumar	(22A85A0524)
S.Bharath Kumar	(21A81A05P7)

TABLE OF CONTENTS

ABSTRACT

Now a days credit card detection is most frequently occurring problem in this world. This is mostly due to online transactions. Credit card fraud generally happens when the card was stolen for any of the unauthorized purposes. To detect the fraudulent activities the credit card fraud detection system was introduced. The main aim of project focus on machine learning algorithms. The algorithms that are used in this project are Random Forest, ANN(Artificial Neural Networks),Naive Bayes. Misclassified data can be another major issue, as not every fraudulent transaction is caught and reported. To tackle this problem we need a system that can abort the transaction if it finds fishy. The main requirement is the past data and suitable algorithms.

Keywords: Random Forest, ANN, Naive Bayes, Machine learning

CHAPTER-1
INTRODUCTION

1.1 Introduction:

For both financial institutions and customers, credit card theft is a common and expensive problem. Machine learning approaches have been used to address this issue, with the XGBoost algorithm emerging as a potent tool for credit card fraud detection.

In essence, the collecting and meticulous preparation of past transaction data is the first step in this process. To make the construction and evaluation of the fraud detection model easier, this data is then split into training and testing sets. The generation of pertinent variables from the transaction data, which can help to identify between legal and fraudulent transactions, is a key aspect of feature engineering during this phase.

The fact that legal transactions vastly outnumber fraudulent ones in the data makes it difficult to detect credit card fraud. To solve this problem, a variety of strategies are frequently used to produce a balanced dataset, such as oversampling the minority class (fraudulent transactions) or under sampling the majority class (legal transactions).

The XGBoost algorithm was selected because to its propensity to generate extremely accurate predictions and its effectiveness in handling imbalanced datasets.

On the basis of the particular requirements and restrictions of the task, alternative machine learning methods, such as Random Forest, Logistic Regression, or Neural Networks, can also be taken into consideration.

During the model training phase, the XGBoost model is trained using the training dataset. Using cross-validation methods to avoid overfitting, hyperparameters are adjusted in this stage to optimize the model's performance.

A crucial phase is model evaluation, in which the effectiveness of the XGBoost model is evaluated using different evaluation measures like accuracy, precision, recall, F1-score, and ROC AUC. Metrics like recall (sensitivity) are given special consideration because failing to detect fraudulent transactions can result in large financial losses.

The model is implemented in a production setting where it may continually assess fresh credit card transactions in real-time after being trained and graded. In order to ensure the model's usefulness over time as fraud trends alter and evolve, regular monitoring and maintenance are required.

When potentially fraudulent transactions are discovered, alerting and reporting methods are often developed to prompt notifications or actions. Reports are given to stakeholders so they can learn more about how well the system is working and how well fraud is being detected.

The XGBoost algorithm is a complicated but crucial procedure that uses machine learning to protect financial transactions while detecting credit card fraud. It includes gathering data, preparing it, designing features, addressing data imbalance, training models, evaluating them, adjusting thresholds, deploying them, monitoring them, and reporting the results. To keep ahead of emerging fraud strategies and defend both customers and financial institutions, constant adaptation and development are necessary.

1.2 Objective:

The objective of credit card fraud detection with XGBoost is to swiftly and accurately identify fraudulent transactions, reducing financial losses, protecting customer trust, and adapting to evolving fraud tactics.

CHAPTER-2

LITERATURE SURVEY

The literature review is the stage of software development that is most crucial. This will outline some preliminary research that was conducted by numerous writers on this relevant work, and we'll take some significant papers into account and continue to develop our work.

[1]Chandra Shekhar, Suman , proposed a research article “Credit Card Fraud Detection Using Machine Learning :A Survey”, the overview of several machine learning approaches and algorithms used in credit card fraud detection is provided in this thorough survey study. It covers approaches for feature engineering, model evaluation, and data pretreatment that are often employed in the industry.

[2]Abdullah Mueen, Aihud Pevzner , proposed a research article "Detecting Credit Card Fraud Using Machine Learning: A Mini Review" in this paper provides a brief overview of machine learning methods for detecting credit card fraud. It emphasizes the difficulties presented by unbalanced datasets and discusses methods like ensemble learning and anomaly detection.

[3]Aditi Bang, Nikita Jivani, and Deep Shah , proposed a research article “Credit Card Fraud Detection using Random Forest Algorithm” in this the Random Forest algorithm's use in identifying credit card fraud is explored in this research. The capability of the method to handle unbalanced datasets and achieve high accuracy is emphasized.

[4]Sergio Montazzolli, Rafael C. Barros, and Michael S. H. M. D. Bandeira , proposed a research article "Credit Card Fraud Detection with a Multi-Class Classification" in order to identify various types of fraud, the study offers a multi-class classification approach for credit card fraud detection. To increase accuracy, the authors use machine learning algorithms.

CHAPTER-3

SYSTEM STUDY AND ANAYLSIS

3.1 Problem Statement:

Develop an efficient real-time credit card fraud detection system that reduces false positives and adapts to evolving fraud tactics. Challenges include class imbalance and quick decision-making, with the aim of reducing financial losses and improving customer trust. This project is essential for financial institutions and cardholders to combat credit card fraud effectively.

3.2Existing System:

The existing system use's the Random forest algorithm , being a bagging model, it generates decision trees and calculates their predictions in parallel.

3.3 Limitations of Existing System:

- The main disadvantage of this algorithm is that due to presence of a large number of trees ,the algorithm can become quite slow and ineffective for real-time predictions.
- Random Forest struggles when there are many more legitimate transactions than fraudulent ones, potentially missing fraud cases.
- It's not easy to understand why Random Forest makes specific predictions, which can be a problem for transparency and regulation.
- Random Forest doesn't adapt quickly to changing fraud patterns, making it less effective over time.

3.4 Proposed System:

The proposed credit card fraud detection system utilizes the XGBoost algorithm for accurate and efficient fraud identification.

3.5 Advantages:

- XGBoost offers high accuracy, handles complex relationships, addresses imbalanced data, provides feature importance insights, and is adaptable to various fraud detection scenarios, making it a powerful choice for credit card fraud detection.
- XGBoost employs an ensemble learning approach, which combines the predictions of multiple weak models to create a stronger overall model.
- XGBoost models can be periodically retrained with updated data to capture emerging fraud patterns and maintain detection accuracy.

3.6 Functional Requirements:

- It should meet the functional requirements as mentioned in Objectives .
- The system should assess the model's performance using metrics such as precision , recall, F1-score, and AUC-ROC on test data.

3.7 Non-functional Requirements:

- Usability
- Reliability
- Portability
- Performance
- Security

3.8 System Requirements:

3.8.1 Software Requirements:

Languages: Python - 3 Version

Operating System: Windows

IDE: Microsoft Visual Studio Code/Google Colab/ Jupyter

3.8.2 Hardware Requirements:

CPU: Intel Core I5 10500H

Hard disk space: 20 GB or more

Memory: 8 GB RAM

CHAPTER-4
SYSTEM DESIGN

4.1 System Architecture:

A system architecture is the conceptual model that defines the structure, behavior, and more views of a system.

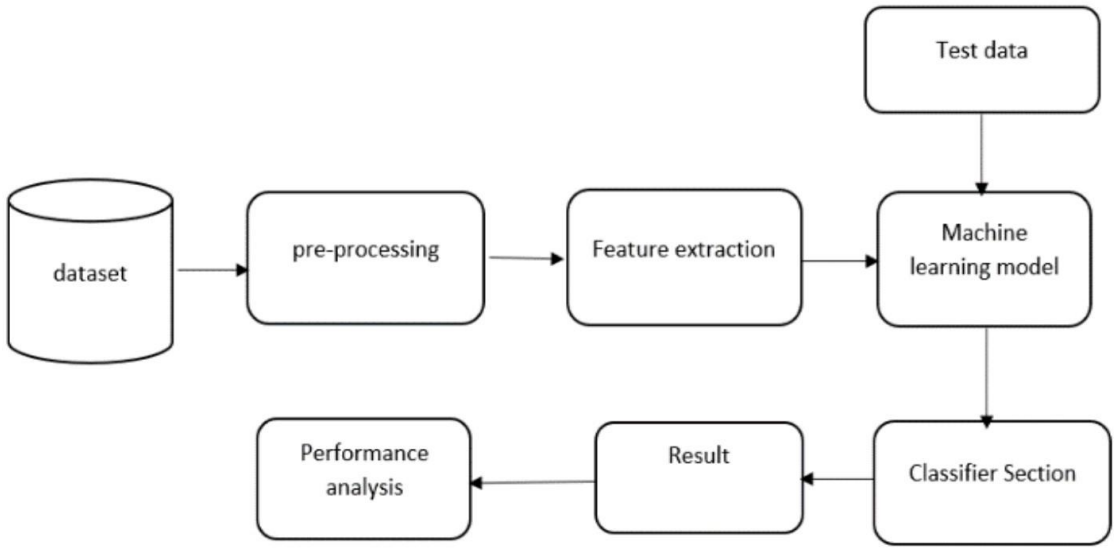


Figure 4.1 System Architecture

4.2 UML Diagrams:

UML is a method for describing the system architecture in detail using the blueprint. UML represents a collection of best engineering practices that has proven successful in the modeling of large and complex systems. The UML is a very important part of developing object-oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects

1.Use Case Diagram

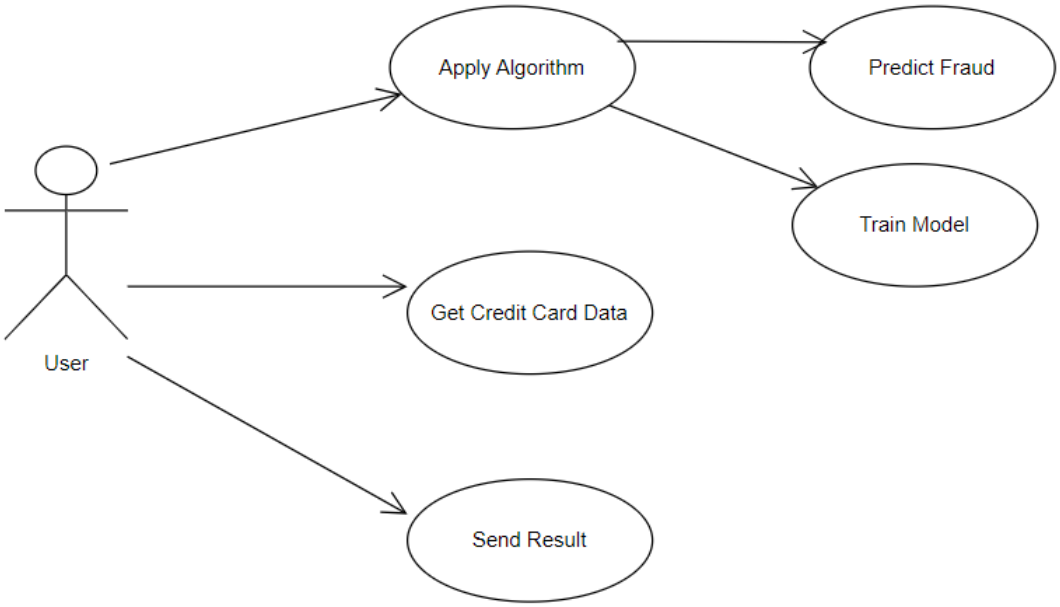
Use case diagram represents the functionality of the system. Use case focus on the behavior of the system from an external point of view. Actors are external entities that interact with the system

Use Cases: A use case describes a sequence of actions that provide something of measurable value to an actor and is drawn as a horizontal ellipse.

Actors: An actor is a person, organization, or external system that plays a role in one or more interactions with the system. Four relationships among use cases are used often in practice.

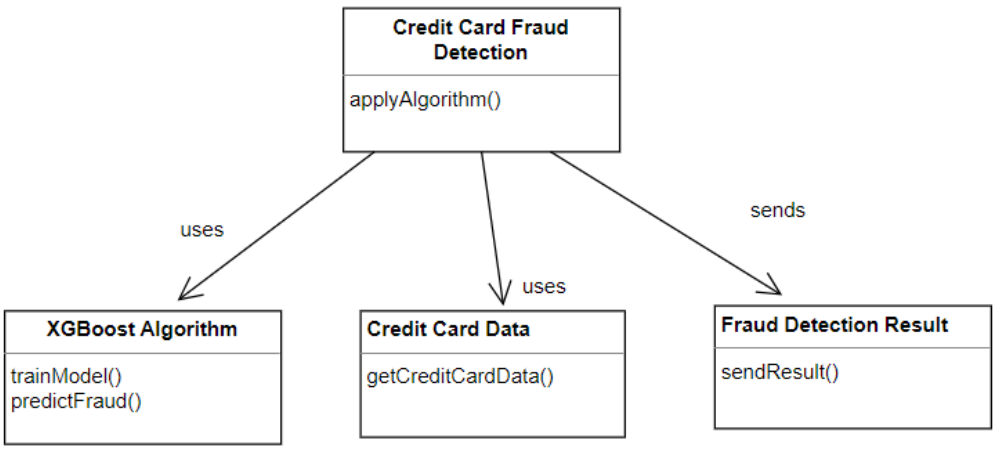
Associations: Associations between actors and use cases are indicated in use case diagrams by solid lines. An association exists whenever an actor is involved with an

interaction described by a use case. Associations are modeled as lines connecting use cases and actors to one another, with an optional arrowhead on one end of the line.



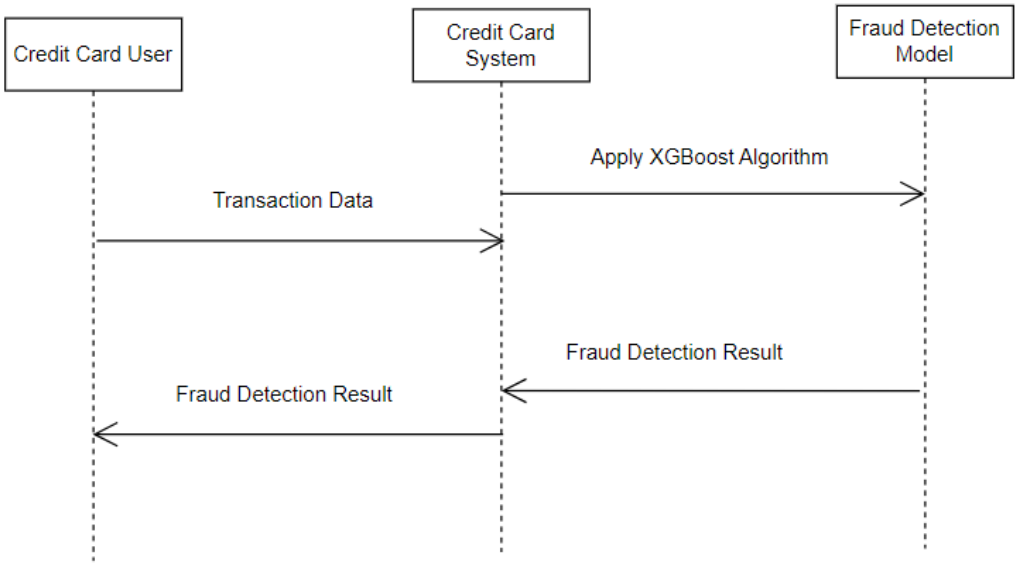
2. Class Diagram

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes.



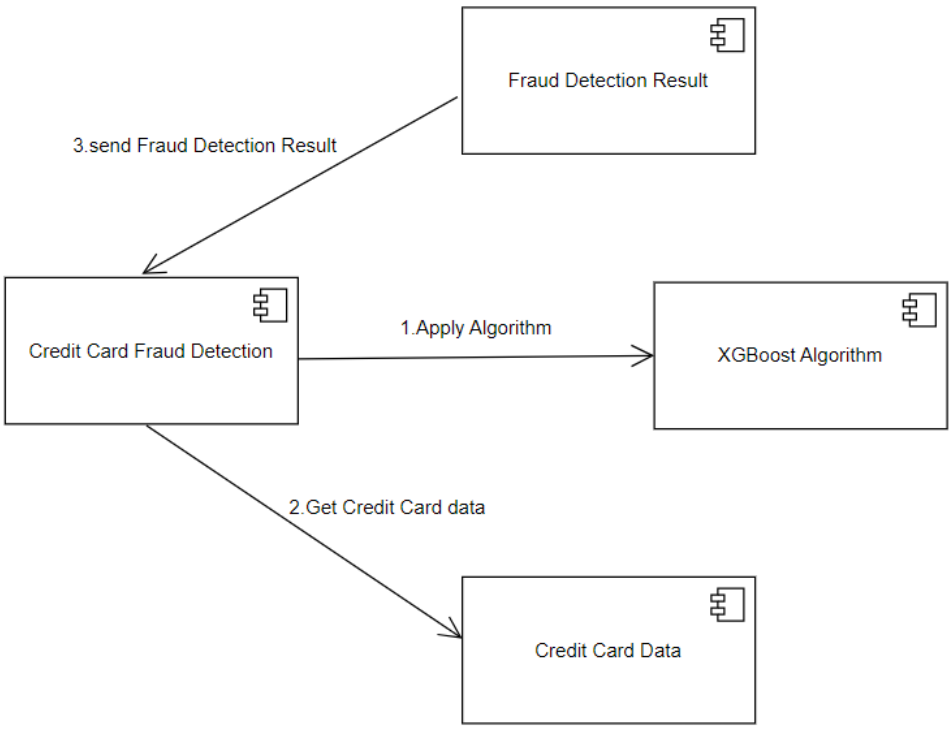
3.Sequence Diagram

A sequence diagram in UML is a visual representation of interactions between objects in a system, showing the order and flow of messages or method calls over time. It provides a dynamic view of how objects collaborate and communicate, making it a valuable tool for understanding system behavior and design. Arrows and lifelines depict the sequence of interactions and the lifespans of objects, respectively.



4.Collabaration Diagram

A collaboration diagram in UML represents interactions between objects or roles in a system, emphasizing collaboration and connections rather than the sequence of messages. It displays objects as participants and shows their associations, roles, and interactions. These diagrams are useful for visualizing complex system architectures and relationships.


Fraud Detection Result
3.send Fraud Detection Result
Credit Card Fraud Detection
1.Apply Algorithm
XGBoost Algorithm
2.Get Credit Card data
Credit Card Data

CHAPTER-5
TECHNOLOGIES

5.1 MACHINE LEARNING

Introduction:

Machine learning is a subset of artificial intelligence (AI) that focuses on the development of algorithms and statistical models that enable computer systems to improve their performance on a specific task through learning from data, without being explicitly programmed. In other words, it's a field of study that teaches computers to learn and make predictions or decisions based on data.

Here's a brief introduction to some fundamental concepts in machine learning:

1.Data: Machine learning relies on data. You need a dataset to train your machine learning model. This dataset typically consists of input features and corresponding output labels. For example, in a spam email classifier, the input features might be the content of the email, and the output label is whether the email is spam or not.

2.Training: During the training phase, a machine learning algorithm uses the provided dataset to learn patterns and relationships within the data. It adjusts its internal parameters to minimize the error in its predictions.

3.Features: Features are the variables or attributes in your dataset that the model uses to make predictions. The quality and relevance of features are crucial for the model's performance.

4.Model: The model is the algorithm that learns from the data. It can be a decision tree, a neural network, a support vector machine, or various other algorithms, depending on the problem you're trying to solve.

5.Supervised Learning: In supervised learning, the model is trained on a labeled dataset, meaning it learns to predict output labels based on input features. Common tasks include classification (assigning a category label) and regression (predicting a continuous value)

6.Unsupervised Learning: Unsupervised learning involves training a model on unlabeled data. The goal is often to find hidden patterns or group similar data points, as in clustering or dimensionality reduction.

7.Validation and Testing: After training, you need to validate the model's performance on a separate dataset (validation data) to tune hyperparameters and avoid overfitting. Testing on yet another dataset (test data) helps assess the model's generalization to new, unseen data.

8.Prediction: Once trained, the model can be used to make predictions or decisions based on new, unseen data.

9.Evaluation: The performance of a machine learning model is assessed using various evaluation metrics, such as accuracy, precision, recall, F1 score, and many others, depending on the specific problem.

10.Deep Learning: Deep learning is a subfield of machine learning that focuses on neural networks with multiple layers. It has been particularly successful in tasks like image and speech recognition, natural language processing, and more.

FEATURES OF MACHINE LEARNING :

- Data-Driven
- Automated Learning
- Generalization
- Adaptability
- Feature Extraction and Engineering
- Pattern Recognition
- Prediction and Classification

CHAPTER-6
IMPLEMENTATION

6.1 IMPLEMENATION STEPS

Implementing a credit card fraud detection system using XGBoost involves several steps. Below are the high-level implementation steps for such a system:

1. Data Collection:

Gather historical credit card transaction data. This dataset should include both genuine and fraudulent transactions.

2. Data Preprocessing:

Clean the data by handling missing values, outliers, and data inconsistencies.

Feature engineering: Create relevant features from the transaction data that can help in fraud detection. Examples include transaction amount, merchant location, transaction time, and more.

3. Data Splitting:

Split the dataset into training and testing sets. The training set is used to train the XGBoost model, while the testing set is used to evaluate its performance.

4. Model Selection:

Choose XGBoost as your machine learning algorithm. You can implement XGBoost using libraries such as XGBoost, scikit-learn, or other machine learning frameworks.

5. Model Training:

Train the XGBoost model on the training dataset. The model will learn to distinguish between genuine and fraudulent transactions.

6. Model Evaluation:

Evaluate the model's performance on the testing dataset using appropriate evaluation metrics such as accuracy, precision, recall, F1 score, and ROC AUC.

7. Hyperparameter Tuning:

Optimize the hyperparameters of the XGBoost model to improve its performance. This may involve using techniques like grid search or random search.

8. Threshold Selection:

Adjust the classification threshold for the model based on the trade-off between false positives and false negatives. This threshold determines when a transaction is classified as potentially fraudulent.

9. Real-time Deployment:

Implement the model in a production environment for real-time credit card transaction processing. This often involves creating an API or service that can receive transaction data and make predictions.

10. Monitoring and Alerts:

Set up continuous monitoring of the model's performance. Implement alerts and notifications to inform relevant parties (e.g., fraud analysts) when potentially fraudulent transactions are detected.

11. Regular Model Updates:

Periodically retrain and update the model with new data to adapt to changing fraud patterns.

12. Fraud Report Generation:

Create reports summarizing detected fraud cases, trends, and activities. These reports can be used for further analysis and decision-making.

13. Compliance and Security:

Ensure that your system complies with relevant regulations and data security standards to protect sensitive customer information.

14. Documentation and Testing:

Document the entire system, including data sources, model architecture, and deployment procedures. Test the system thoroughly to ensure its reliability.

15. User Training:

Train relevant personnel, such as fraud analysts, in using the system and interpreting its output.

16. Scaling and Performance Optimization:

As the system processes more transactions, consider optimizing its performance and scalability to handle increased loads.

17.Maintenance:

Continuously monitor the system's performance, retrain the model as needed, and keep all components up to date.

6.2 XG BOOST ALGORITHM :

XGBoost, which stands for eXtreme Gradient Boosting, is a popular and powerful machine learning algorithm used for supervised learning tasks, particularly in the domains of regression and classification. It is known for its efficiency, accuracy, and versatility. XGBoost is an extension of the gradient boosting algorithm and has gained widespread popularity in data science and machine learning competitions for its effectiveness.

Here's an introduction to XGBoost:

1. Gradient Boosting: XGBoost is a gradient boosting algorithm, which is an ensemble technique that builds a strong predictive model by combining the predictions of multiple

weaker models (typically decision trees).The term "gradient" in gradient boosting refers to the optimization process that reducing the errors made by the previous models.

2. Decision Trees: XGBoost primarily uses decision trees as base models (sometimes called "weak learners" or "stumps"). These trees are combined to create a stronger ensemble model.

3.Regularization: XGBoost incorporates L1 (Lasso) and L2 (Ridge) regularization techniques to control model complexity and prevent overfitting. This helps in improving the generalization ability of the model.

4. Parallel and Distributed Computing: XGBoost is designed to take advantage of parallel and distributed computing, making it highly efficient and capable of handling large datasets. It can be run on multi-core CPUs and distributed computing environments.

5. Handling Missing Data: XGBoost has built-in support for handling missing data, which reduces the need for data preprocessing.

6.Cross-Validation: Cross-validation is an essential part of model training and hyperparameter tuning. XGBoost has tools for k-fold cross-validation, helping you evaluate model performance more robustly.

7. Feature Importance: XGBoost provides a feature importance score, which helps identify the most critical features in your dataset, enabling better feature selection and engineering.

8. Early Stopping: XGBoost allows early stopping during the training process, which means the model can automatically stop training when it no longer improves on a validation dataset. This helps prevent overfitting and saves computation time.

9. High Predictive Accuracy: XGBoost is renowned for its high predictive accuracy and is often used in machine learning competitions where model performance is critical.

10. Support for Multiple Objective Functions: XGBoost supports various objective functions, making it adaptable to different types of supervised learning tasks, including regression, classification, ranking, and more.

In summary, XGBoost is a versatile and efficient gradient boosting algorithm that excels in various machine learning tasks. It's highly regarded for its predictive performance and has been widely adopted in both academia and industry for a wide range of applications, including predictive modeling, anomaly detection, and recommendation systems.

CHAPTER-7
TESTING

7.1 Test Objective

Testing credit card fraud detection systems is critical to ensure their effectiveness in identifying and preventing fraudulent transactions while minimizing false positives. Test objectives for credit card fraud detection typically include:

Accuracy Testing: Verify that the system accurately identifies fraudulent transactions.

False Positive Rate Testing: Evaluate the system's ability to minimize false positives (legitimate transactions wrongly flagged as fraudulent).

True Positive Rate Testing: Confirm that the system correctly identifies true positives (fraudulent transactions).

Performance Testing: Assess the system's ability to process a high volume of transactions efficiently.

Threshold Testing: Validate that the chosen classification threshold effectively balances between false positives and false negatives.

Data Quality Testing: Ensure that the system can handle a wide range of transaction data, including missing values, outliers, and anomalies.

Model Validation: Test the model's performance and validity, including assessing its predictive power and generalization capabilities.

Scenario Testing: Evaluate the system's response to various fraud scenarios, such as card-not-present transactions, card-present transactions, and more.

Speed and Latency Testing: Measure the time it takes for the system to process and classify transactions in real-time.

7.2 Test Case Design

Designing test cases for a credit card fraud detection system using XGBoost involves creating scenarios to validate the system's performance, accuracy, and reliability. Below are examples of test cases that cover various aspects of the system:

1.Test Case: Basic Transaction Classification

Objective: Verify that the system can correctly classify a basic, legitimate transaction.

Input: A known genuine transaction.

Expected Output: The system should classify the transaction as non-fraudulent (not a fraud).

2.Test Case: Basic Fraudulent Transaction Classification

Objective: Verify that the system can correctly classify a basic fraudulent transaction.

Input: A known fraudulent transaction.

Expected Output: The system should classify the transaction as fraudulent.

3.Test Case: Threshold Adjustment

Objective: Ensure that changing the classification threshold has the expected impact on the system's behavior.

Input: Vary the classification threshold value.

Expected Output: The system should classify transactions differently based on the adjusted threshold.

4.Test Case: Handling Missing Data

Objective: Verify that the system can handle transactions with missing data gracefully.

Input: A transaction with missing or incomplete data.

Expected Output: The system should either process the transaction or flag it for manual review, depending on the configuration.

5.Test Case: Outlier Detection

Objective: Ensure that the system can detect and handle transactions with unusual or outlier characteristics.

Input: A transaction with outlier features.

Expected Output: The system should either classify the transaction correctly or flag it for manual review.

7.3 Unit Testing

Unit testing is essential to ensure the individual components and functions of your credit card fraud detection system using XGBoost work as intended. Here are some key unit testing scenarios and test cases for this system:

1.Data Preprocessing Unit Tests:

i. Test Case: Data Cleaning

Objective: Ensure that data cleaning functions handle missing or invalid data.

Input: A dataset with missing values and outliers.

Expected Output: Verify that missing values are imputed or removed, and outliers are addressed.

ii. Test Case: Feature Engineering

Objective: Validate that feature engineering functions create relevant features for the model.

Input: Raw transaction data.

Expected Output: Check that new features are correctly generated and contribute to the model's performance.

2.XGBoost Model Unit Tests:

i. Test Case: Model Training

Objective: Confirm that the XGBoost model can be trained with training data.

Input: Training dataset.

Expected Output: Verify that the model is successfully trained without errors.

ii. Test Case: Model Hyperparameter Tuning

Objective: Validate that hyperparameter tuning improves model performance.

Input: Different sets of hyperparameters.

Expected Output: Ensure that the model performs better with optimized hyperparameters.

3.Threshold Adjustment Unit Tests:

i. Test Case: Threshold Change

Objective: Ensure that changing the classification threshold affects model behavior.

Input: Different threshold values.

Expected Output: Verify that the model's classification results change accordingly.

4.Model Evaluation Unit Tests:

i. Test Case: Evaluation Metrics Calculation

Objective: Validate that the system calculates evaluation metrics accurately.

Input: Model predictions and ground truth labels.

Expected Output: Verify that metrics such as accuracy, precision, recall, F1-score, and ROC AUC are calculated correctly.

7.4 Integration Testing

Integration testing for a credit card fraud detection system using XGBoost involves verifying the interactions and interfaces between different components and services. Here are some integration testing scenarios and test cases for such a system:

1.Data Integration Testing:

i. Test Case: Data Ingestion

Objective: Verify that the system can correctly ingest data from various sources, such as databases, external APIs, or data streams.

Input: Sample data from different sources.

Expected Output: Ensure that data is successfully imported into the system.

ii. Test Case: Data Transformation

Objective: Validate that data is transformed and prepared for model input correctly.

Input: Raw data from different sources.

Expected Output: Verify that data preprocessing and feature engineering functions work as expected.

2.Model Integration Testing:

i. Test Case: Model Deployment

Objective: Confirm that the trained XGBoost model can be successfully deployed within the system.

Input: The trained model.

Expected Output: Ensure that the model is operational and can accept transaction data for classification.

ii. Test Case: Model Evaluation Integration

Objective: Validate that the system can evaluate the model's performance using real data.

Input: Real transaction data and model predictions.

Expected Output: Verify that evaluation metrics are calculated accurately.

3.Alerting and Notification Integration Testing:

i. Test Case: Notification Delivery

Objective: Ensure that the system can send alerts and notifications to the appropriate recipients.

Input: Simulated fraud detection scenario.

Expected Output: Confirm that notifications are sent as expected (e.g., email alerts, SMS messages).

4.Reporting Integration Testing:

i. Test Case: Report Generation Integration

Objective: Validate the generation of fraud reports using real data.

Input: Detected fraud cases and transaction details.

Expected Output: Confirm that comprehensive fraud reports are generated accurately.

CHAPTER-8
SCREENSHOTS

8.1 Coding

```
[1] from google.colab import drive
    drive.mount('/content/drive')

    Mounted at /content/drive
```

```
import pandas as pd
import numpy as np
import xgboost as xgb
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix,roc_curve, roc_auc_score, auc
from sklearn.preprocessing import LabelEncoder
import seaborn as sns
```

```
[4] data = pd.read_csv('/content/drive/MyDrive/set.csv')
```

```
[ ] data.shape

    (284807, 31)
```

```
[ ] data.head()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28
0	0.0	-1.359807	-0.072781	2.536347	1.378155	-0.338321	0.462388	0.239599	0.098698	0.363787	...	-0.018307	0.277838	-0.110474	0.066928	0.128539	-0.189115	0.133558	-0.021053
1	0.0	1.191857	0.266151	0.166480	0.448154	0.060018	-0.082361	-0.078803	0.085102	-0.255425	...	-0.225775	-0.638672	0.101288	-0.339846	0.167170	0.125895	-0.008983	0.014724
2	1.0	-1.358354	-1.340163	1.773209	0.379780	-0.503198	1.800499	0.791461	0.247676	-1.514654	...	0.247998	0.771679	0.909412	-0.689281	-0.327642	-0.139097	-0.055353	-0.059752

Connected to Python 3 Google Compute Engine backend

```
data.tail()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27
284802	172786.0	-11.881118	10.071785	-9.834783	-2.066656	-5.364473	-2.606837	-4.918215	7.305334	1.914428	...	0.213454	0.111864	1.014480	-0.509348	1.436807	0.250034	0.943651
284803	172787.0	-0.732789	-0.055080	2.035030	-0.738589	0.868229	1.058415	0.024330	0.294869	0.584800	...	0.214205	0.924384	0.012463	-1.016226	-0.606624	-0.395255	0.068472
284804	172788.0	1.919565	-0.301254	-3.249640	-0.557828	2.630515	3.031260	-0.296827	0.708417	0.432454	...	0.232045	0.578229	-0.037501	0.640134	0.265745	-0.087371	0.004455
284805	172788.0	-0.240440	0.530483	0.702510	0.689799	-0.377961	0.623708	-0.686180	0.679145	0.392087	...	0.265245	0.800049	-0.163298	0.123205	-0.569159	0.546668	0.108821
284806	172792.0	-0.533413	-0.189733	0.703337	-0.506271	-0.012546	-0.649617	1.577006	-0.414650	0.486180	...	0.261057	0.643078	0.376777	0.008797	-0.473649	-0.818267	-0.002415

5 rows × 31 columns

```
[ ] data.describe()
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22
count	284807.000000	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	2.848070e+05	...	2.848070e+05	2.848070e+05
mean	94813.859575	1.168375e-15	3.416908e-16	-1.379537e-15	2.074095e-15	9.604066e-16	1.487313e-15	-5.556467e-16	1.213481e-16	-2.406331e-15	...	1.654067e-16	-3.568593e-16
std	47488.145955	1.958696e+00	1.651309e+00	1.516255e+00	1.415869e+00	1.380247e+00	1.332271e+00	1.237094e+00	1.194353e+00	1.098632e+00	...	7.345240e-01	7.257016e-01
min	0.000000	-5.640751e+01	-7.271573e+01	-4.832559e+01	-5.683171e+00	-1.137433e+02	-2.616051e+01	-4.355724e+01	-7.321672e+01	-1.343407e+01	...	-3.483038e+01	-1.093314e+01
25%	54201.500000	-9.203734e-01	-5.985499e-01	-8.903648e-01	-8.486401e-01	-6.915971e-01	-7.682956e-01	-5.540759e-01	-2.086297e-01	-6.430976e-01	...	-2.283949e-01	-5.423504e-01

```
[ ] data.columns

    Index(['Time', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10',
           'V11', 'V12', 'V13', 'V14', 'V15', 'V16', 'V17', 'V18', 'V19', 'V20',
           'V21', 'V22', 'V23', 'V24', 'V25', 'V26', 'V27', 'V28', 'Amount',
           'Class'],
          dtype='object')
```

```
data.nunique()

Time      124592
V1        275663
V2        275663
V3        275663
V4        275663
V5        275663
V6        275663
V7        275663
V8        275663
V9        275663
V10       275663
V11       275663
V12       275663
V13       275663
V14       275663
V15       275663
V16       275663
V17       275663
V18       275663
V19       275663
V20       275663
V21       275663
V22       275663
```

```
[6] X = data.drop('Class', axis=1)
    Y = data['Class']
```

```
[7] legit = data[data.Class==0]
    fraud = data[data['Class']==1]
```

```
fraud['Class']

541       1
623       1
4920      1
6108      1
6329      1
         ..
279863    1
280143    1
280149    1
281144    1
281674    1
Name: Class, Length: 492, dtype: int64
```

+ Code + Text

```
[8] X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=42)
```

```
[9] from sklearn.naive_bayes import GaussianNB
    naive_bayes_classifier = GaussianNB()
    naive_bayes_classifier.fit(X_train, Y_train)
```

```
[10] y_pred = naive_bayes_classifier.predict(X_test)

[12] accuracy = accuracy_score(Y_test, y_pred)
     print(f"Accuracy: {accuracy * 100:.2f}%")

     Accuracy: 99.30%

[13] conf_matrix = confusion_matrix(Y_test, y_pred)
     print("Confusion Matrix:")
     print(conf_matrix)

     Confusion Matrix:
     [[56502   362]
      [   36    62]]

[14] class_report = classification_report(Y_test, y_pred)
     print("Classification Report:")
     print(class_report)

     Classification Report:
                   precision    recall  f1-score   support

                0       1.00      0.99      1.00     56864
                1       0.15      0.63      0.24        98

         accuracy                           0.99     56962
        macro avg       0.57      0.81      0.62     56962
     weighted avg       1.00      0.99      1.00     56962
```

```
[15] from sklearn.ensemble import RandomForestClassifier
     random_forest_classifier = RandomForestClassifier(n_estimators=100, random_state=42)
     random_forest_classifier.fit(X_train, Y_train)

          RandomForestClassifier
     RandomForestClassifier(random_state=42)

     y_pred = random_forest_classifier.predict(X_test)

[18] accuracy = accuracy_score(Y_test, y_pred)
     print(f"Accuracy: {accuracy * 100:.2f}%")

     Accuracy: 99.96%

[20] conf_matrix = confusion_matrix(Y_test, y_pred)
     print("Confusion Matrix:")
     print(conf_matrix)

     Confusion Matrix:
     [[56862     2]
      [   23    75]]

     class_report = classification_report(Y_test, y_pred)
     print("Classification Report:")
     print(class_report)
```

```
[ ] model = xgb.XGBClassifier()

[ ] valid_indices = ~np.isnan(Y_train)
    X_train = X_train[valid_indices]
    Y_train = Y_train[valid_indices]

[ ] params = {
        'objective': 'binary:logistic',
        'max_depth': 3,
        'learning_rate': 0.1,
        'n_estimators': 100,
        'eval_metric': 'logloss'
    }

    model = xgb.XGBClassifier(**params)
    model.fit(X_train, Y_train)

                                XGBClassifier
    XGBClassifier(base_score=None, booster=None, callbacks=None,
                  colsample_bylevel=None, colsample_bynode=None,
                  colsample_bytree=None, device=None, early_stopping_rounds=None,
                  enable_categorical=False, eval_metric='logloss',
                  feature_types=None, gamma=None, grow_policy=None,
                  importance_type=None, interaction_constraints=None,
                  learning_rate=0.1, max_bin=None, max_cat_threshold=None,
```

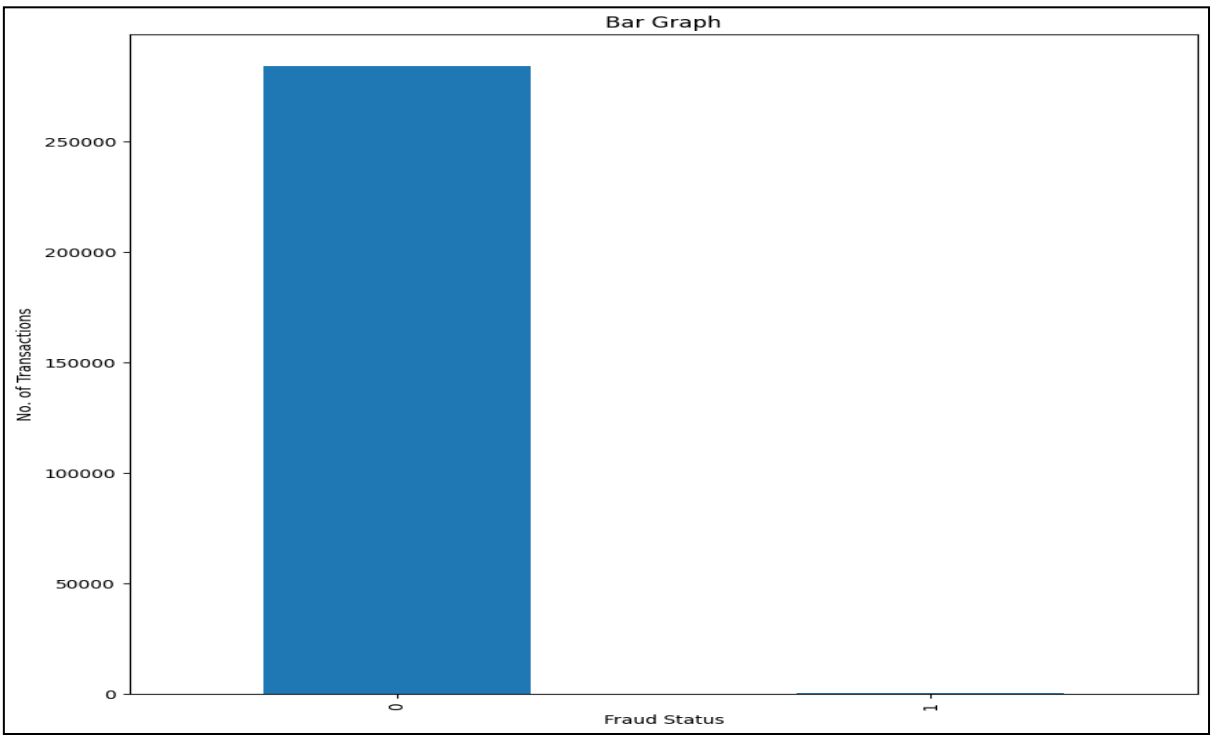
```
[ ] y_pred = model.predict_proba(X_test)[:, 1]

[ ] fpr, tpr, thresholds = roc_curve(Y_test, y_pred)
    roc_auc = auc(fpr, tpr)

    plt.figure(figsize=(8, 6))
    plt.plot(fpr, tpr, color='darkorange', lw=2, label='ROC curve (area = {:.2f})'.format(roc_auc))
    plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver Operating Characteristic (ROC) Curve')
    plt.legend(loc='lower right')
    plt.show()
```

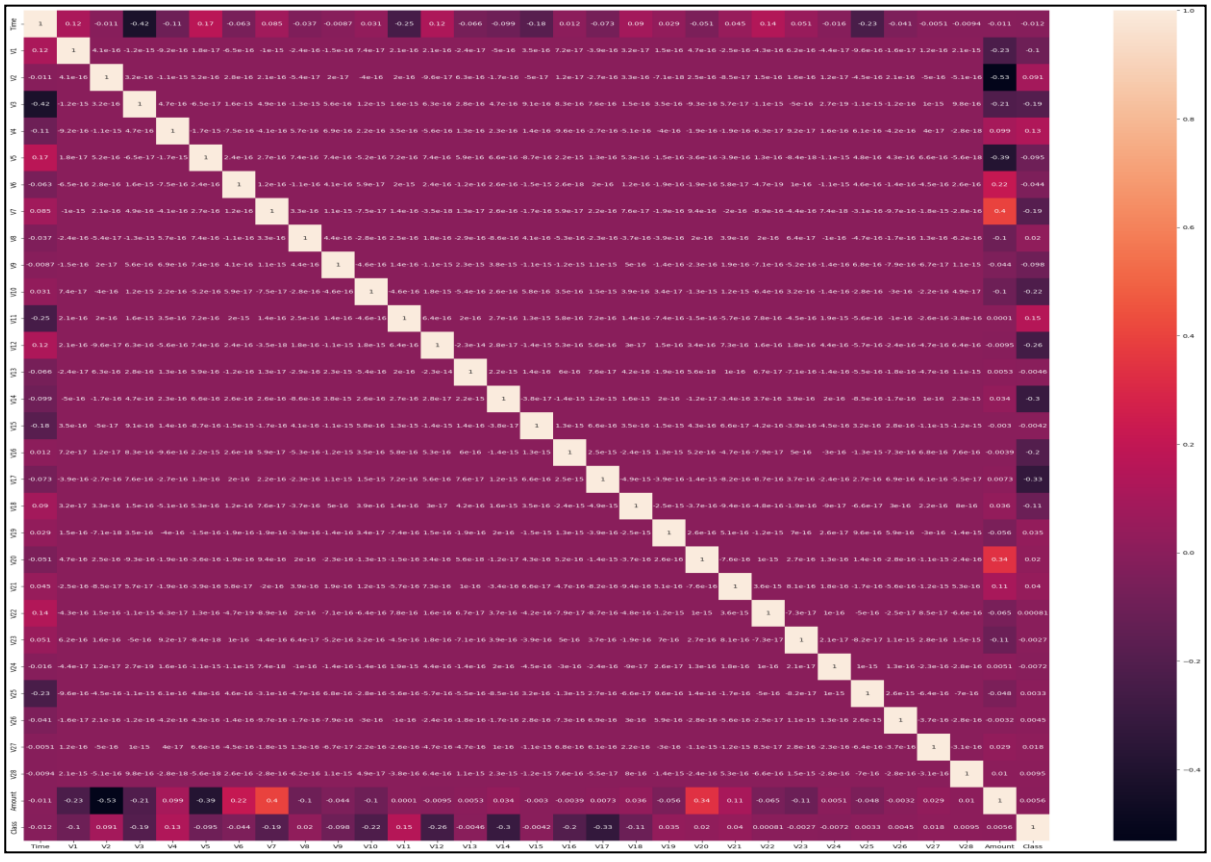
```
[ ] data.Amount[data.Class == 1].max()

    2125.87

[ ] y_pred = model.predict(X_test)

    cols = list(range(31))
    fig, ax = plt.subplots(figsize=(30,30))
    corelation = data.iloc[:,cols].corr()
    sns.heatmap(corelation, xticklabels = corelation.columns, yticklabels = corelation.columns, annot = True)
    plt.savefig('Correlation matrix range all.png')
```

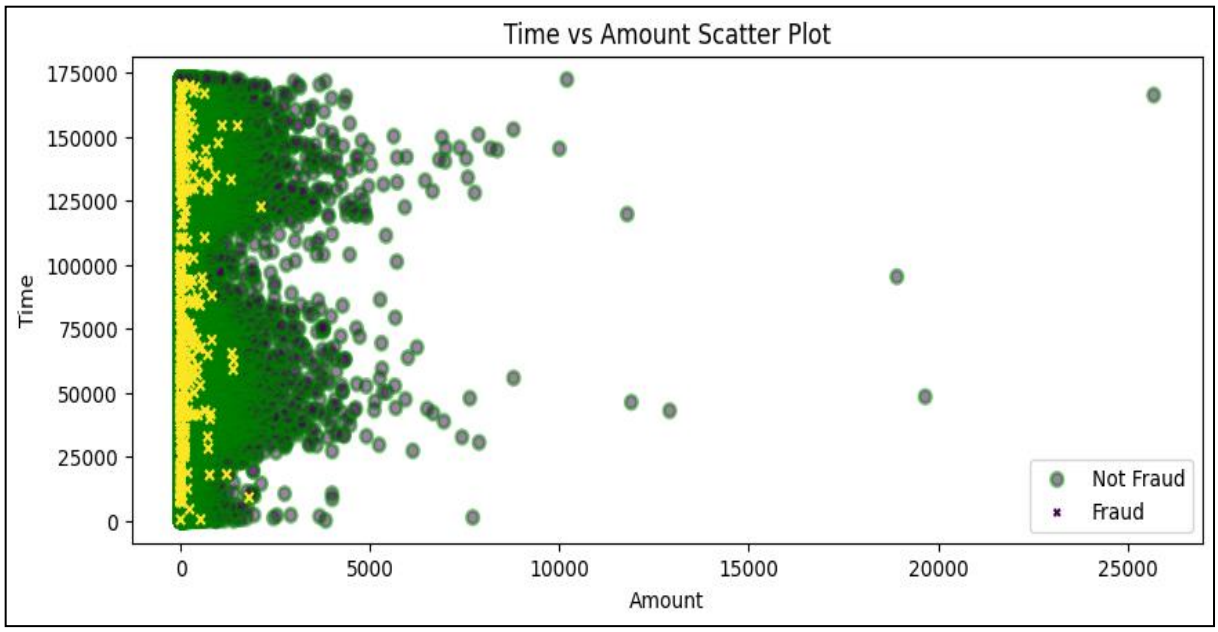
```
[ ] fig, ax = plt.subplots(figsize=(10,4))
    plt.scatter(data.Amount, data.Time, c=data.Class,alpha = 0.5,edgecolor='green',linewidth= 2, label = 'Not Fraud')
    plt.scatter(data.Amount, data.Time, marker = 'x', s = 20*data.Class , c = data.Class, label = 'Fraud')
    plt.title('Time vs Amount Scatter Plot')
    plt.xlabel('Amount')
    plt.ylabel('Time')
    plt.legend()
    plt.show()
```

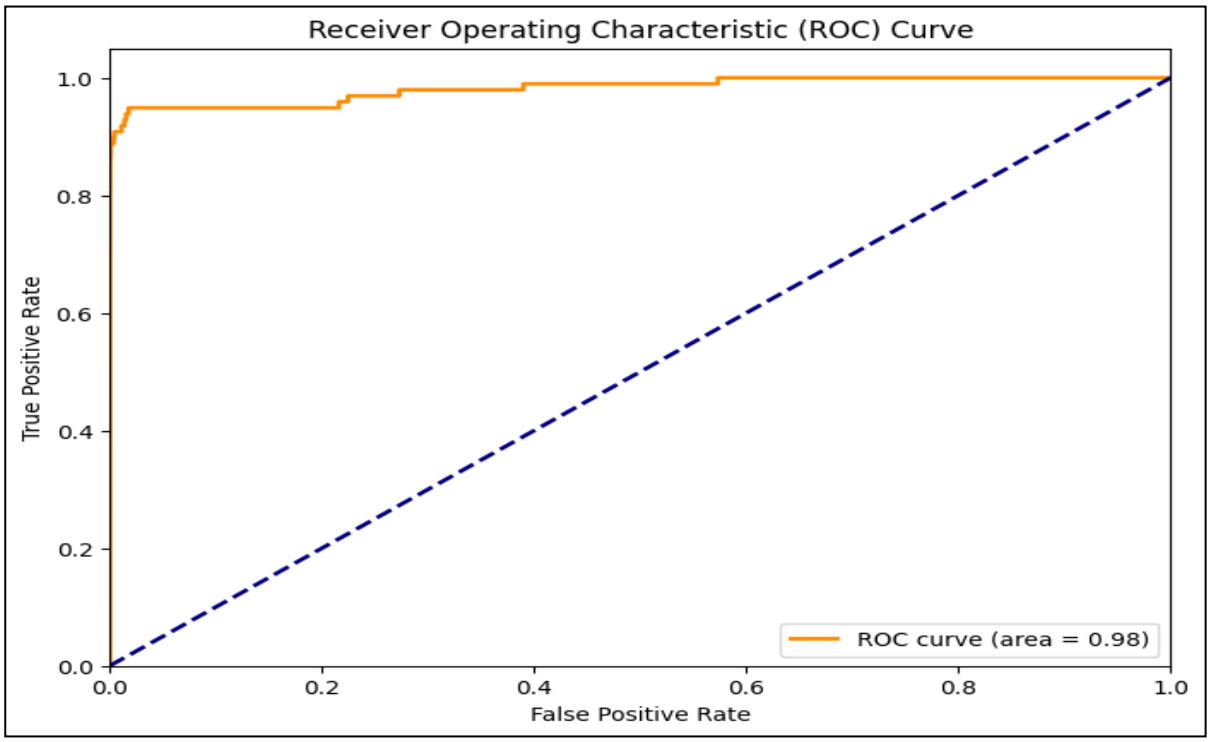
```
[ ] accuracy = accuracy_score(Y_test, y_pred)
    conf_matrix = confusion_matrix(Y_test, y_pred)
    classification_rep = classification_report(Y_test, y_pred)

    print(f"Accuracy: {accuracy}")
    print("Confusion Matrix:\n", conf_matrix)
    print("Classification Report:\n", classification_rep)

    Accuracy: 0.9995611109160493
    Confusion Matrix:
     [[56859     5]
     [   20    78]]
    Classification Report:
                  precision    recall  f1-score   support

               0       1.00      1.00      1.00     56864
               1       0.94      0.80      0.86        98

        accuracy                           1.00     56962
       macro avg       0.97      0.90      0.93     56962
    weighted avg       1.00      1.00      1.00     56962

[ ] for i in range(len(y_pred)):
        if y_pred[i] == 1:
            print(f"Transaction {i + 1}: Fraud")
        else:
            print(f"Transaction {i + 1}: Not Fraud")
```

8.2 Graphs


Bar Graph
No. of Transactions
250000
200000
150000
100000
50000
0
0
1
Fraud Status




Time vs Amount Scatter Plot
Time
Amount
Not Fraud
Fraud
0
25000
50000
75000
100000
125000
150000
175000
0
5000
10000
15000
20000
25000


Receiver Operating Characteristic (ROC) Curve
True Positive Rate
False Positive Rate
ROC curve (area = 0.98)
0.0
0.2
0.4
0.6
0.8
1.0

CHAPTER-9

CONCLUSION AND FUTURE WORK

9.1 Conclusion

Finally, the credit card fraud detection model leveraging XGBoost demonstrates its capacity to accurately spot fraudulent transactions with a 99.9% accuracy rate, which is rather remarkable.

While high accuracy is encouraging, it's important to take into account additional metrics like precision, recall, and F1-score to fully assess the model.

Additionally, sustaining such high accuracy in real-world situations with changing fraud trends necessitates ongoing monitoring, updates, and threshold adjustment to find the ideal balance between false positives and false negatives, ensuring strong and dependable fraud detection.

9.2 Future Work

Our vision extends beyond the current capabilities, and we are committed to developing a system that can continually evolve and address new challenges in the realm of financial security.

By understanding how people usually use their money, we could find strange behavior and stop fraud.

We might also make it send real-time alerts to people to keep their money safe. It could work with other systems and even help people in other countries.

As fraud tactics change over time, the project will evolve to meet these challenges and stay at the forefront of fraud detection.

References

[1] “Credit Card Fraud Detection: A Realistic Modelling and a Novel Learning Strategy” published by IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, VOL. 29, NO. 8, AUGUST 2018.

[2] Xuan S, Liu G, Li Z, Zheng L, Wang S, Jiang C. Random forest for credit card fraud detection. In: 2018 IEEE 15th international conference on networking, sensing and control (ICNSC). IEEE; 2018.

[3] Adepoju, O., Wosowei, J., lawte, S., & Jaiman, H. (2019). Comparative evaluation of credit card fraud detection using machine learning techniques. 2019 Global Conference for Advancement in Technology (GCAT).

[4] Jemima Jebaseeli T, Venkatesan R, Ramalakshmi K. Fraud detection for credit card transactions using random forest algorithm. Singapore: Springer; 2020.

[5] Saheed, Y. K., Hambali, M. A., Arowolo, M. O., & Olasupo, Y. A. (2020). Application of ga feature selection on Naive Bayes, random forest and SVM for credit card fraud detection. 2020 International Conference on Decision Aid Sciences and Application (DASA).

[6] Gupta, A., Lohani, M. C., & Manchanda, M. (2021). Financial fraud detection using naive Bayes algorithm in highly imbalance data set. Journal of Discrete Mathematical Sciences and Cryptography