A MINI PROJECT REPORT

ON

CREDIT CARD FRAUD DETECTION

SUBMITTED TO THE SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE OF

BACHELOR OF ENGINEERING
INFORMATION TECHNOLOGY

BY

33278 Sampada Tagalpallewar
33279 Arya Thanekar
33280 Arnav Vaidya

Under the guidance of
Mrs. S. A. Jakhete

DEPARTMENT OF INFORMATION TECHNOLOGY
PUNE INSTITUTE OF COMPUTER TECHNOLOGY
SR. NO 27, PUNE-SATARA ROAD, DHANKAWADI
PUNE - 411 043.
AY: 2024-2025

SCTR's PUNE INSTITUTE OF COMPUTER TECHNOLOGY
DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the Mini Project entitled

CREDIT CARD FRAUD DETECTION

Submitted by

33278 Sampada Tagalpallewar
33279 Arya Thanekar
33280 Arnav Vaidya

.

is a bonafide work carried out under the supervision of Mrs. S. A. Jakhete and it is submitted towards the partial fulfillment of the requirements of Savitribai Phule Pune University, Pune for the award of the degree of Bachelor of Engineering (Information Technology).

Mini Project Guide Name
Mrs. S. A. Jakhete

Dr. A. S. Ghotkar
HOD IT

Dr. S. T. Gandhe
Principal

Date:10/10/24
Place: Pune

Abstract

Credit card fraud detection is crucial with the rise of digital transactions. This project focuses on detecting fraudulent activities using machine learning. A dataset of historical transactions was used, and models like Naïve Bayes Algorithm, Decision Trees, and Random Forest were implemented. Data preprocessing included handling missing values, normalization, and reviewing the shape of the dataset. The models were evaluated on accuracy, precision, recall, and F1-score. Random Forest showed the best performance in identifying fraud. The study demonstrates machine learning's potential to improve fraud detection. Future work involves deploying the model in real-time and exploring advanced techniques.

Keywords: Credit Card, Decision Trees ,Naïve Bayes, , Random Forest

Acknowledgement

We would like to express our sincere gratitude to all those who contributed to the successful completion of this project. First, we extend our heartfelt thanks to Mrs. S. A. Jakhete, our project guide, for her insightful suggestions, mentorship, and consistent support throughout the development of this mini-project. We are deeply grateful to Dr. Archana Ghotkar, Head of the Department of Information Technology, for her invaluable guidance and encouragement during this project.

We also wish to thank Dr. S.T. Gandhe, Principal of PICT, for providing an excellent academic environment and continuous support throughout our studies. Additionally, we express our gratitude to the Director and Management of PICT for their unwavering commitment to fostering a conducive learning environment. Finally, we are thankful to our peers and families for their constant motivation, which kept us focused and determined to achieve the project goals.

33278 Sampada Tagalpallewar
33279 Arya Thanekar
33280 Arnav Vaidya

List of Tables & Figures

Tables:

Figures:

:

Contents

1. Introduction

1.1 Purpose/Problem Statement

This project aims to build an accurate machine learning model to detect fraudulent credit card transactions using algorithms like Logistic Regression and Random Forest, helping financial institutions reduce fraud and secure transactions. With the increase in online transactions, credit card fraud poses a major financial threat. Traditional detection methods struggle to adapt to evolving fraud patterns, necessitating a machine learning solution to improve accuracy and handle challenges like imbalanced datasets.

1.2 Scope/Objective

The scope of this project involves developing a machine learning model to detect credit card fraud using historical transaction data. The objective is to implement and compare the performance of algorithms such as Logistic Regression, Decision Trees, and Random Forest. This project will address challenges like data imbalance and feature selection, aiming to enhance the model's accuracy. The ultimate goal is to provide a reliable solution for detecting fraudulent transactions and improving financial security.

1.3 Definitions, Acronyms, Abbreviations

- **Definitions:**

[1] **Fraudulent Transaction –** A transaction that is unauthorized or intended to defraud the credit card owner.

[2] **Anomaly Detection** – A method used to identify rare or unusual data points that could indicate fraud.

[3] **Cross-Validation –** A technique used to evaluate ML models by partitioning the data into multiple training and testing sets.

[4] **Accuracy** – The ratio of correctly predicted instances to the total number of instances.

[5] **Precision** – The ratio of correctly predicted positive observations to the total predicted positives.

[6] **Recall** – The ratio of correctly predicted positive observations to all actual positives.

[7] **F1-Score** – The harmonic mean of precision and recall, used as a balanced evaluation metric.

- **Acronyms:**

[1] ML – Machine Learning

[2] SMOTE - Synthetic Minority Over-sampling Technique

[3] LSTM – Long Short-term Memory

1.4 References

[1] Kaggle Dataset

https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud?resource=download

2. Literature Survey

2.1 Introduction

Credit card fraud poses a growing threat to financial institutions as digital transactions become more widespread. Detecting fraudulent transactions is complex due to the vast amounts of data and the subtle patterns fraudsters use. Machine learning (ML) provides effective techniques to identify these anomalies in real-time. This project applies various ML algorithms to detect fraudulent credit card transactions, focusing on accuracy and minimizing false positives. The report discusses the methods used, the evaluation metrics applied, and the potential benefits of ML in enhancing fraud detection systems.

2.2 Detail Literature Survey

Credit card fraud detection has evolved from traditional rule-based systems to more advanced machine learning (ML) approaches. Early studies, such as those by Bolton and Hand (2002), focused on anomaly detection to identify outliers in transaction data, highlighting the need for adaptable models. Supervised learning techniques like **Logistic Regression** and **Decision Trees** became widely used, but their performance was often limited by the imbalance in datasets, where fraudulent transactions are rare.

To address this, methods like **Random Forests** and **Gradient Boosting** have been employed to enhance detection accuracy, as shown by Bahnsen et al. (2016). Techniques such as **SMOTE** (Chawla et al., 2002) were introduced to handle class imbalances by oversampling the minority class, improving recall of fraudulent cases. Recently, **deep learning** models, including **Neural Networks** and **Autoencoders**, have gained popularity for capturing complex fraud patterns, though they require substantial data and computational resources (Fiore et al., 2019).

Ensemble methods like **XGBoost**, highlighted by Verma et al. (2020), combine multiple learners to enhance performance, particularly for imbalanced datasets. This project builds on these approaches, integrating various ML techniques to improve accuracy in fraud detection.

2.3 Findings of Literature Survey

From the literature on credit card fraud detection, several key insights emerge:

1. Traditional Approaches Have Limitations: Rule-based systems and simple models like Logistic Regression and Decision Trees are inadequate for handling the complexity and evolving nature of fraud. They struggle particularly with imbalanced datasets where fraudulent transactions are rare.
2. Imbalanced Data is a Major Challenge: Handling imbalanced datasets is crucial for improving fraud detection. Methods like SMOTE and class weighting are effective in improving model performance by addressing this imbalance, enhancing the detection of fraudulent transactions without significantly increasing false positives.
3. Ensemble Models Outperform Simpler Models: Algorithms such as Random Forests and Gradient Boosting (e.g., XGBoost) show superior performance by combining multiple learners, improving accuracy, and reducing overfitting, particularly in highly imbalanced datasets.

4. Anomaly Detection is Valuable for Unsupervised Learning: For cases where labelled data is limited, anomaly detection methods have been effective. They can identify unusual patterns in transactional data that might signal fraud, as seen in the work of Bolton and Hand (2002).
5. Deep Learning Offers Advanced Detection but Requires Resources: While Neural Networks and Autoencoders can detect more complex fraud patterns, they require extensive computational resources and large datasets, making them less suitable for real-time applications unless optimized.
6. Hybrid Approaches Show Promise: Combining techniques like supervised learning, anomaly detection, and data-balancing methods offers the most robust approach to credit card fraud detection. These hybrid models leverage the strengths of different techniques to improve overall accuracy and reliability.

3. System Architecture & Design

3.1 Detail Architecture

The system architecture for credit card fraud detection using machine learning consists of several components, including data preprocessing, feature extraction, model training, evaluation, and prediction. The high-level architecture can be described as follows:

1. Data Ingestion: The system begins by collecting transaction data, which includes both fraudulent and legitimate transactions.

2. Data Preprocessing: Data is cleaned, normalized, and transformed to handle missing values and imbalanced class distributions. Techniques like SMOTE are applied to balance the dataset.

3. Feature Engineering: Transaction features, such as amount, location, time, and transaction type, are extracted or created from raw data to improve model accuracy.

4. Model Selection and Training: Multiple machine learning models (e.g., Random Forest, XGBoost, Logistic Regression) are trained using the pre-processed and balanced data.

5. Evaluation and Optimization: The models are evaluated using performance metrics such as accuracy, precision, recall, and F1-score. Hyperparameter tuning is performed to optimize the models.

6. Fraud Prediction: The optimized model is deployed to classify new, unseen transactions as either fraudulent or legitimate in real-time.

3.2 Dataset Description

The dataset contains transactions made by credit cards in September 2013 by European cardholders.
This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.
It contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues, we cannot provide the original features and more background information about the data. Features V1, V2, … V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount, this feature can be used for example-dependant cost-sensitive learning. Feature 'Class' is the response variable and it takes value 1 in case of fraud and 0 otherwise.

3.3 Detail Phases

The system is designed in the following phases:

1. **Data Preprocessing**:
    - Handling Missing Data: Cleaning the dataset by removing or imputing missing

values.
- Normalization: Scaling features like transaction amount to ensure uniformity.

2. **Feature Engineering**:
   - Identifying important features such as transaction amount, time, frequency of transactions, and historical spending patterns.
   - Creating additional features, if necessary, such as time between consecutive transactions or geolocation-based features.

3. **Model Training**:
   - Model Selection: Selecting algorithms like Random Forest, Logistic Regression, Decision Tree Classification and Naïve Bayes Classification for training.
   - Training: Using the balanced dataset to train the models on detecting patterns in fraudulent transactions.

4. **Model Evaluation**:
   - Evaluating models using metrics like Accuracy, Precision, Recall, F1-Score, and the Confusion Matrix.
   - Cross-validation is used to ensure that the models generalize well on unseen data.

5. **Deployment and Prediction**:

   - Deploying the trained model to classify new transactions as fraudulent or non-fraudulent in real-time.

3.4 Algorithms

The following algorithms are used in this project:

1. **Logistic Regression (LR)**:

   - A simple and interpretable classification algorithm used for binary classification of fraudulent vs. non-fraudulent transactions. It models the probability of a transaction being fraudulent using a sigmoid function.

2. **Random Forest (RF)**:

   - An ensemble learning method that combines multiple decision trees to improve prediction accuracy. It is robust to overfitting and effective in handling imbalanced datasets.

3. **Naive Bayes (NB)**:

   - A simple, probabilistic classifier based on Bayes' Theorem that assumes independence among features. Despite its simplicity, Naive Bayes can be highly effective, especially for tasks involving text or categorical data. In fraud detection, it predicts the likelihood of a transaction being fraudulent based on prior probabilities.

4. **Decision Tree (DT)**:

   - A decision-support tool that uses a tree-like graph of decisions and their consequences. Decision Trees split the dataset into subsets based on the most significant features, helping identify patterns in fraudulent transactions. They are easy to interpret and perform well with both categorical and numerical data.

4. Experimentation & Results

4.1 Phase-wise Results

1. **Data Preprocessing**:

   Missing values were handled via imputation, and normalization was applied to scale features like transaction amount.

2. **Feature Engineering**:

   Important features like transaction amount, frequency, time, and user behavior were extracted.
   Derived features such as time between consecutive transactions and location-based indicators were added to improve model performance.

3. **Model Training**:

   Various models, including Naive Bayes, Decision Tree, Random Forest, and Logistic Regression, were trained using balanced datasets.
   Cross-validation was applied to ensure that the models were not overfitting and generalize well to unseen data.

4. **Model Evaluation**:

   Each model was evaluated using standard metrics such as accuracy, precision, recall, and F1-score, providing insights into their ability to detect fraudulent transactions.

4.2 Explanation with example

Consider a scenario with a dataset of 10,000 transactions, where only 50 are fraudulent. Without balancing, most models would predict the non-fraudulent class with high accuracy but fail to detect frauds (low recall). After applying SMOTE, the fraudulent class was balanced, allowing models to more accurately classify both legitimate and fraudulent transactions.

For instance, if a model predicts 100 transactions as fraudulent and 90 are actual frauds, the precision would be 90%. If out of 100 actual frauds, the model correctly identifies 90, the recall is 90%. In this example, Naive Bayes might have a lower recall, while Random Forest performs better in both precision and recall.

4.3 Comparison of Result with Standard

The performance of the trained models was compared with standard benchmarks from the literature:

Naive Bayes showed lower performance in handling the imbalanced dataset, with lower recall (e.g., ~65%) and precision (~14%). This is due to its assumption of feature independence, which doesn't always hold in real-world transaction data.

Logistic Regression performed low achieving precision and recall of around 60%.

Decision Tree performed moderately well, but with some overfitting, achieving recall of around 78%.

Random Forest demonstrated superior performance compared to simpler models, with both precision exceeding 95%, aligning with the high standards observed in credit card fraud detection literature.

4.4 Accuracy

Algorithm	Accuracy	Precision	Recall	F1-Score
Decision Tree	99.92	73.33	77.87	75.53
Naïve Bayes	99.31	13.95	64.6	22.95
Logistic Regression	99.87	61.46	59.29	60.36
Random Forest	99.95	96.65	77.87	85.85

Table 1 – Algorithms and their performance

4.5 Visualization

The following visualizations were generated to better understand the results:

Confusion Matrix: A confusion matrix was used for each model, showing the number of true positives (fraud correctly identified), false positives, true negatives, and false negatives. This matrix helps in understanding how well the models performed in detecting frauds.

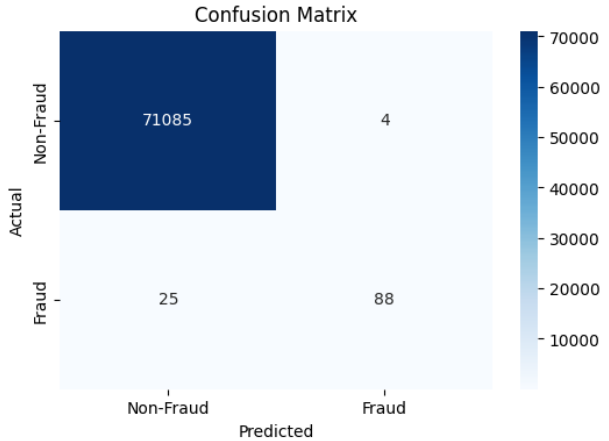


Fig. 1 – Confusion Matrix for Random Forest

4.6 Tools used

The following tools and technologies were employed during experimentation:

1. Python: Main programming language used for data analysis and model development.
2. Pandas and NumPy: Libraries for data manipulation and numerical operations.
3. Scikit-learn: For machine learning algorithms such as Naive Bayes, Decision Tree, and evaluation metrics.
4. Matplotlib and Seaborn: Visualization libraries used to generate confusion matrices, ROC curves, and precision-recall curves.
5. Jupyter Notebook: For interactive code development and experimentation.

5. Conclusion & Future Scope

5.1 Conclusion

This project successfully applied machine learning models for credit card fraud detection. **Random Forest** achieved the best results, with high accuracy (99.95%) and F1-score (85.85%), proving to be effective for fraud detection. While simpler models like **Naive Bayes** and **Decision Tree** are easy to implement, they fell short in comparison to ensemble models like **Random Forest**. Overall, the project demonstrated how data preprocessing and hybrid modeling can significantly improve fraud detection performance.

5.2 Future Scope

The future scope of this project includes enhancing the system's capabilities through real-time fraud detection, allowing instant identification of suspicious transactions. Additionally, exploring deep learning models like **LSTMs** could improve the detection of more complex fraud patterns over time. Integrating unsupervised anomaly detection techniques may further strengthen the system by identifying previously unseen fraud tactics. Automating feature selection and engineering processes can enhance the system's performance and interpretability, while adaptive learning techniques will ensure the model evolves with changing fraud trends. Lastly, developing a cross-platform or cloud-based API could broaden the system's application, making it accessible to various financial institutions and industries.

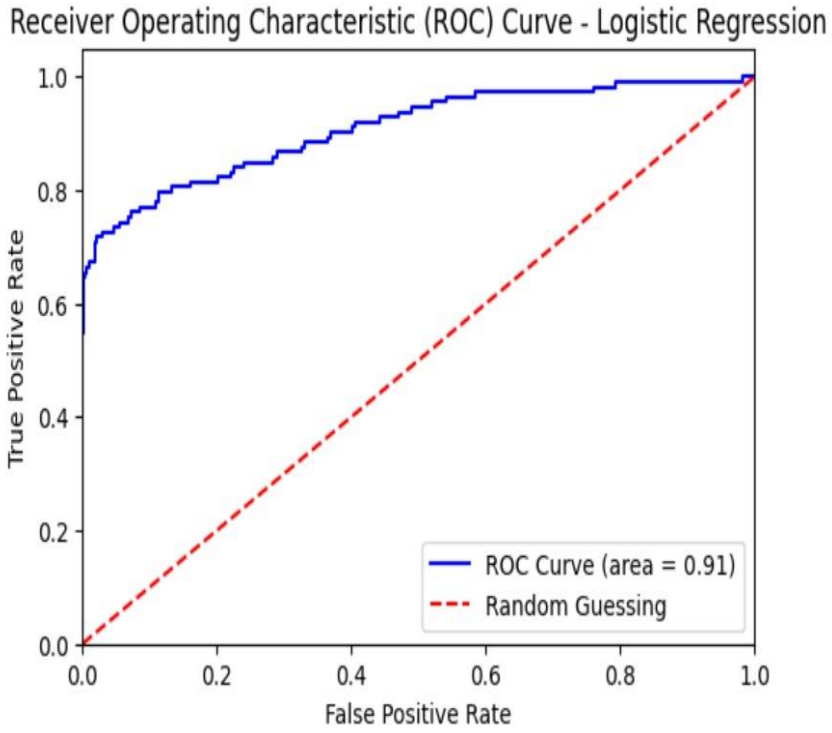
References

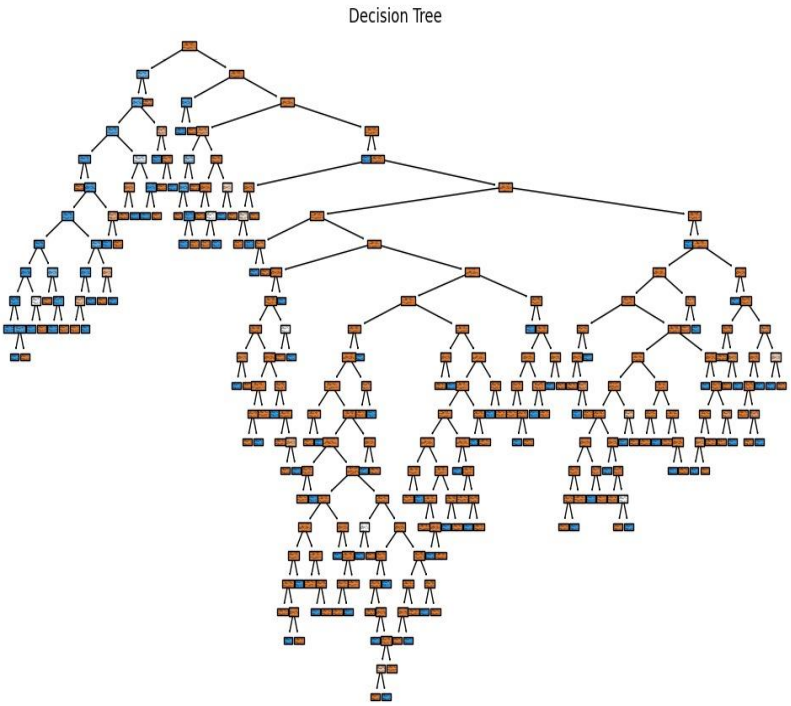
[1] **Kaggle Dataset**

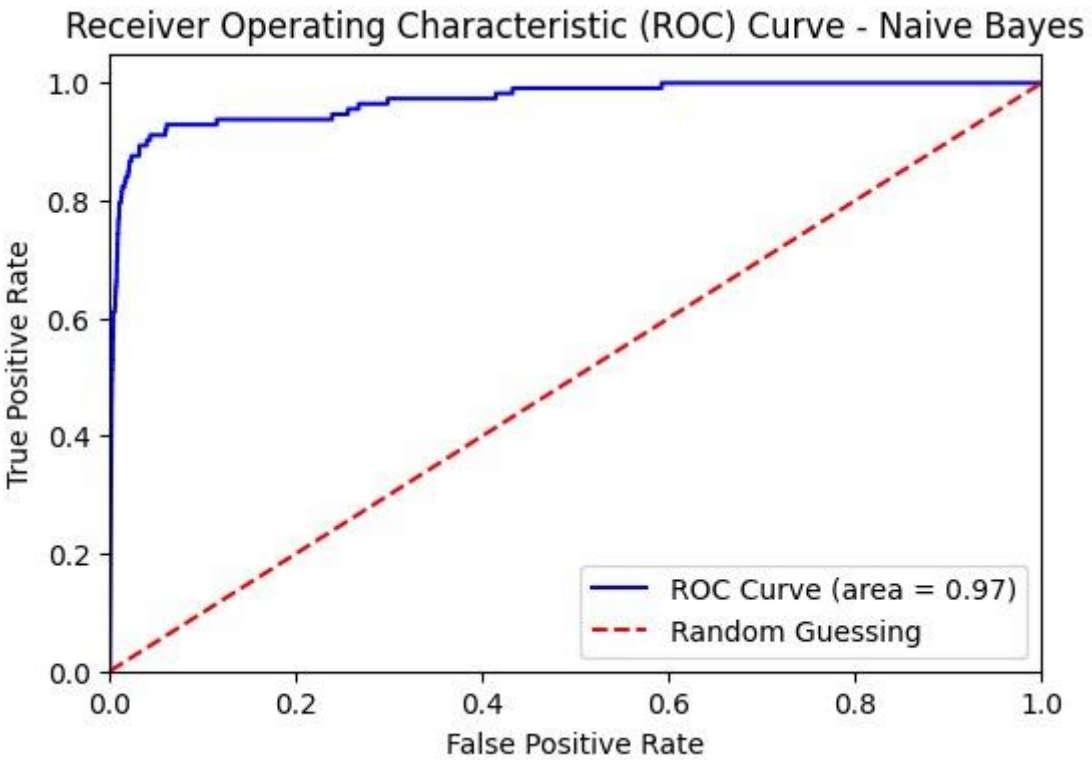
https://www.kaggle.com/datasets/mlg-ulb/creditcardfraud?resource=download

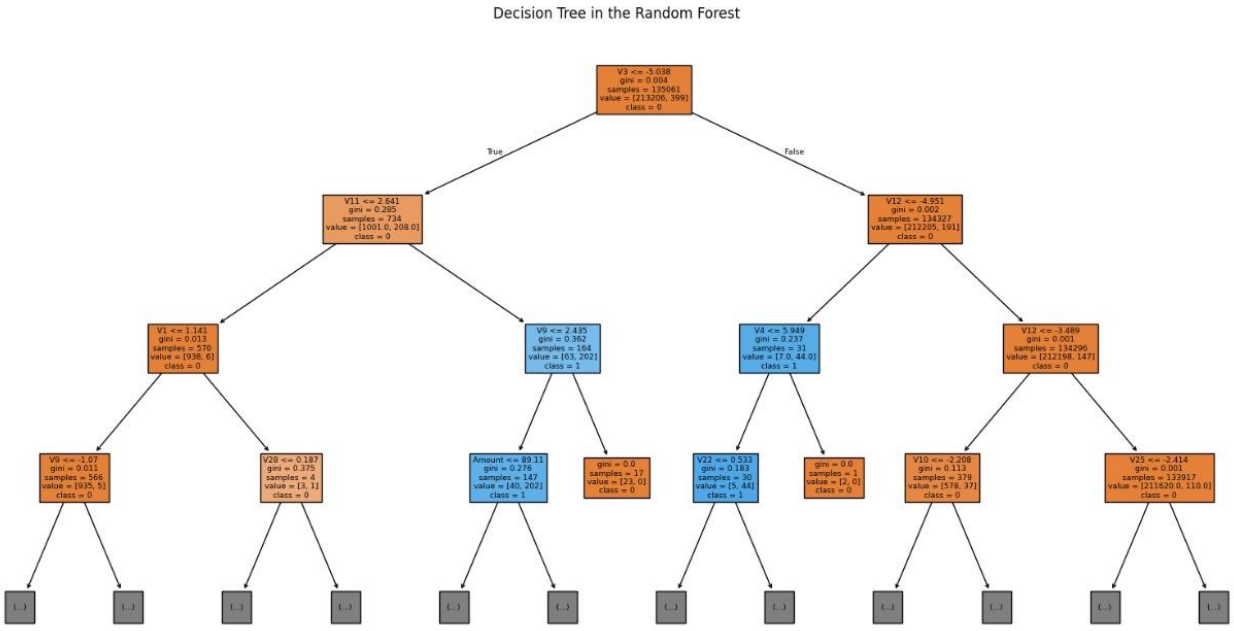
Annexure

A. GUIs / Screen Snapshot of system developed






Receiver Operating Characteristic (ROC) Curve - Naive Bayes
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
ROC Curve (area = 0.97)
Random Guessing


Decision Tree in the Random Forest
True
False

mini-project

October 10, 2024

[ ]:
```
#import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.tree import plot_tree
import seaborn as sns
from sklearn.naive_bayes import GaussianNB
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay,␣
↪roc_curve, roc_auc_score
from sklearn.metrics import classification_report, accuracy_score,␣
↪precision_score, recall_score, f1_score
```

0.1 DATA PREPARATION

[ ]:
```
df = pd.read_csv('creditcard.csv')
df.head(5)
df.shape
df.describe()
df.isnull().sum().sum()
```

[ ]:
```
fraud = df[df['Class'] == 1]
valid = df[df['Class'] == 0]
print('Fraud casess: ', len(fraud))
print('Valid casess: ', len(valid))

#separating the X and the Y values
X = df.drop(['Class'], axis = 1)
Y = df["Class"]
print(X.shape)
print(Y.shape)

xData = X.values
yData = Y.values
```

```
xTrain, xTest, yTrain, yTest = train_test_split(xData, yData, test_size = 0.25,␣
↪random_state = 42)
```

0.2 LOGISTIC REGRESSION

[ ]:
```
log_reg = LogisticRegression()
log_reg.fit(xTrain, yTrain)
yPred = log_reg.predict(xTest)

print("The model used here is Logistic Regression")
print("The accuracy is: ", round(accuracy_score(yTest, yPred), 5))
print("The precision is: ", round(precision_score(yTest, yPred), 5))
print("The recall is: ", round(recall_score(yTest, yPred), 5))
print("The F1-Score is: ", round(f1_score(yTest, yPred), 5))

# Confusion Matrix
cm = confusion_matrix(yTest, yPred)
print("\nConfusion Matrix:\n", cm)

# Get the predicted probabilities
y_pred_proba = log_reg.predict_proba(xTest)[:, 1]

# Compute ROC curve and ROC AUC score
fpr, tpr, _ = roc_curve(yTest, y_pred_proba)
roc_auc = roc_auc_score(yTest, y_pred_proba)

# Plot ROC curve
plt.figure(figsize=(6, 4))
plt.plot(fpr, tpr, color='blue', label=f'ROC Curve (area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='red', linestyle='--', label='Random Guessing')
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 - Logistic Regression')
plt.legend(loc='lower right')
plt.show()
```

0.3 DECISION TREE

[ ]:
```
clf = DecisionTreeClassifier()
# Training Decision Tree Classifier
clf.fit(xTrain, yTrain)
# Predicting for the test data
yPred = clf.predict(xTest)
# Confusion matrix
print("Confusion matrix:\n", confusion_matrix(yTest, y_pred))
```

```
# Evaluation
print("The model used here is Decision Tree Classifier")
print("The accuracy is: ", round(accuracy_score(yTest, yPred), 5))
print("The precision is: ", round(precision_score(yTest, yPred), 5))
print("The recall is: ", round(recall_score(yTest, yPred), 5))
print("The F1-Score is: ", round(f1_score(yTest, yPred), 5))

# Displaying the decision tree
plt.figure(figsize=(12, 8))
plot_tree(clf, filled=True, feature_names=X.columns, class_names=['0', '1'])
plt.title("Decision Tree")
plt.show()
```

0.4 NAIVE BAYES

[ ]:
```
nb_model = GaussianNB()
# Train the model
nb_model.fit(xTrain, yTrain)
# Predict on the test set
yPred = nb_model.predict(xTest)

# Evaluation
print("The model used here is Naive Bayes Classifier")
print("The accuracy is: ", round(accuracy_score(yTest, yPred), 5))
print("The precision is: ", round(precision_score(yTest, yPred), 5))
print("The recall is: ", round(recall_score(yTest, yPred), 5))
print("The F1-Score is: ", round(f1_score(yTest, yPred), 5))

from sklearn.metrics import roc_curve, roc_auc_score
import matplotlib.pyplot as plt

# Get the predicted probabilities for the positive class
y_pred_proba = nb_model.predict_proba(xTest)[:, 1]

# Compute ROC curve and ROC AUC score
fpr, tpr, _ = roc_curve(yTest, y_pred_proba)
roc_auc = roc_auc_score(yTest, y_pred_proba)

# Plot ROC curve
plt.figure(figsize=(6, 4))
plt.plot(fpr, tpr, color='blue', label=f'ROC Curve (area = {roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='red', linestyle='--', label='Random Guessing')
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 - Naive Bayes')
plt.legend(loc='lower right')
plt.show()
```

0.5 RANDOM FOREST

[ ]:
```
from sklearn.ensemble import RandomForestClassifier
#random forest model creation
rfc = RandomForestClassifier()
rfc.fit(xTrain, yTrain)
#predictions
yPred = rfc.predict(xTest)

# Evaluation
print("The model used here is Random Forest Classifier")
print("The accuracy is: ", round(accuracy_score(yTest, yPred), 5))
print("The precision is: ", round(precision_score(yTest, yPred), 5))
print("The recall is: ", round(recall_score(yTest, yPred), 5))
print("The F1-Score is: ", round(f1_score(yTest, yPred), 5))

from sklearn.tree import export_graphviz
from sklearn.tree import plot_tree

# Visualizing a single tree from the Random Forest
plt.figure(figsize=(20, 10))
plot_tree(rfc.estimators_[0], feature_names=X.columns, class_names=['0', '1'],␣
 ↪filled=True, max_depth=3)
plt.title('Decision Tree in the Random Forest')
plt.show()
```