A project report on

CREDIT CARD FRAUD DETECTION

Submitted in partial fulfilment for the award of the degree of

COMPUTER SCIENCE AND ENGINEERING IN COLLABORATION WITH VIRTUSA

By

21MIC7147- GUNDA KARTHEEK

21MIC7171- MOHAMMAD NUZHAT KULSUM

21MIC7191- SHAIK AFIFA ALIYA

AMARAVATI

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

July,2024

CREDIT CARD FRAUD DETECTION

Submitted in partial fulfillment for the award of the degree of

COMPUTER SCIENCE AND ENGINEERING IN COLLABORATION WITH VIRTUSA

by

21MIC7147- GUNDA KARTHEEK

21MIC7171- MOHAMMAD NUZHAT KULSUM

21MIC7191- SHAIK AFIFA ALIYA

AMARAVATI

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

July 2024

CERTIFICATE

This is to certify that the thesis entitled “**CREDIT CARD FRAUD DETECTION”** submitted by Md Nuzhat Kulsum(21MIC7171), Shaik Afifa Aliya(21MIC7191), Gunda Karthee (21MIC7147) VIT-AP, for the award of the Summer Internship for the bonafide work carried out by him/her under my supervision. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The Project report fulfils the requirements and regulations of VIT-AP and in my opinion meets the necessary standards for submission.

DR. GOKUL YENDURI
Signature of the Guide

ABSTRACT

As the internet becomes more accessible, businesses are expanding their online offerings. Additionally, since e-commerce websites have grown in popularity, people and companies in the financial industry are increasingly depending on internet marketing managers to run their operations. The number of credit card fraud cases has risen due to the growing popularity of online banking and shopping. The systematic operation of the current fraud detection system might be interrupted by fraudsters using any means possible to overcome this problem using Machine learning algorithms and statistical methods have been used in credit card fraud detection to work past these restrictions. These methods are predicated on the analysis of important variables, like the customer's transaction history and account information, in addition to transaction-related data, like the transaction amount, location, and time by identifying and filtering fraudulent activity in real time, this research aims to reduce fraud manipulation by creating an effective fraud detection system through the use machine learning techniques that adapt to changing patterns of customer behaviour. The techniques include Logistic Regression, Random Forest, Decision tree and Explainable Artificial Intelligence (XAI). These models have demonstrated positive outcomes in identifying fraudulent transactions by learning data patterns and improving fraud detection efficiency. Overall, credit card fraud detection is an important field of research in the financial industry, with an opportunity for improving fraud detection rates and minimize financial losses.

ACKNOWLEDGEMENT

It is my pleasure to express with deep sense of gratitude to **MR.GOKUL YENDURI**, **SCOPE, VIT-AP** for constant guidance, continual encouragement, understanding; more than all, he taught me patience in my endeavour. My association is not confined to academics only, but it is a great opportunity on my part of work with an intellectual and expert in the field of Department of Software and System Engineering.

I would like to express my gratitude to Dr. G. Viswanathan, Sankar Viswanathan, S. V. Kota Reddy, and Dr. CH. Pradeep Reddy, SCOPE, for providing with an environment to work in and for his inspiration during the tenure of the course. In jubilant mood I express ingeniously my whole-hearted thanks to Dr. Nagaraju Devarakonda, Assistant Professor, VIT-AP, Program Chair of MTCSE, all teaching staff and members working as limbs of our university for their not-self-centred enthusiasm coupled with timely encouragements showered on me with zeal, which prompted the acquirement of the requisite knowledge to finalize my course study successfully.

I would like to thank my parents for their support. It is indeed a pleasure to thank my friends who persuaded and encouraged me to take up and complete this task. At last, but not least, I express my gratitude and appreciation to all those who have helped me directly or indirectly toward the successful completion of this project.

Place: AMRAVATI
Date:31-07-2024

Name of the student
MD NUZHAT KULSUM
SHAIK AFIFA ALIYA
GUNDA KARTHEE

List of Figures

List of Tables

CHAPTER-1

INTRODUCTION

1.1 PURPOSE

The project aims to create a machine learning-based credit card fraud detection system using Random Forest, Decision Tree, and Logistic Regression. Accurately identifying fraudulent transactions while reducing false positives is the goal. Furthermore, the work utilizes Explainable AI (XAI) methodologies to make its decision-making procedure understandable and transparent, thus encouraging overall assurance.

1.2 SCOPE

The project's goal is to build an effective system that can identify credit card fraud. it monitors transactions and look for patterns to identify suspicious activity using machine learning. Possible fraud will be identified through pattern, and reports will help investigators quickly look into incidents. It providing a dependable solution that improves transaction security and lowers financial losses for clients and banks is the ultimate objective.

1.3 TECHNOLOGIES TO BE USED

- Dataset (Kaggle)
- Coding (VISUAL STUDIOS)
- Machine Learning:

  1) Logistic Regression.

  2) Random Forest.

  3) Decision Tree.
- Explainable AI

  1)SHAP

1.4 OVERVIEW

Since fraud involving credit cards directly affects banks as well as clients, it is an important issue for the world of finance. Machine learning (ML) algorithms provide strong capabilities to detect fraudulent transactions with a high level of accuracy in order to address this problem. In order to identify fraudulent activity in credit card transactions, this project focuses on using methods based on machine learning, such as decision trees, random forests, and logistic regression. Each of these algorithms has a unique benefit, such as the clarity and simplicity of Logistic Regression, the durability and capacity for collective learning of Random Forest, and the decision-making ways that are visible in Decision Tree models. For the purpose to improve these algorithms' transparency and guarantee that users understand the decision-making process, we use Explainable AI (XAI) methods. This helps us to understand the problem and also to improve the identification of fraud while maintaining trust and transparency in the financial systems by combining these modern analysis techniques**.**

CHAPTER-2

BACKGROUND

2.1 GOAL OF PROPOSED SYSTEM.

Improving the accuracy and efficiency of detecting fraudulent transactions while reducing inaccurate results and negative results is the main objective of the suggested credit card fraud detection system. The system attempts to create reliable predictions that can accurately distinguish between authentic and fraudulent activity by utilizing machine learning algorithms like random forest, logistic regression, decision tree. these algorithms' transparency and guarantee that users understand the decision-making process, we use Explainable AI (XAI) methods.

2.2 ALGORITHM DESCRIPTION

2.2.1 Logistic Regression.

Logistic regression is a supervised machine learning algorithm used for classification tasks where the goal is to predict the probability that an instance belongs to a given class or not. Logistic regression is a statistical algorithm which analyse the relationship between two data factors, Binary classification problems are solved with logistic regression. Using this method, input data are linearly merged with weights or coefficients for predicting the output value. The general logistic regression equation is Y = e^(b0+b1x)/1+e^(bo+b1x). In this case, y is the expected value of the output, b1 is the coefficient, and b0 is the value of the bias. A logistic regression produces a binary value, which can only be 0 or 1.

Logistic regression is used for predicting trends, causal analysis, and outcome prediction.

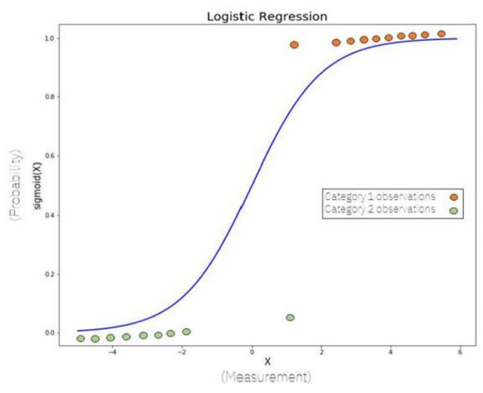


Figure of 2.2.1 Logistic Regression.

2.2.2 RANDOM FOREST

A machine learning technique called Random Forest makes use of several decision trees to increase reliability of predictions. A single, effective result is produced by Random Forest by combining the output of several trees. This approach works well to handle complicated data and preventing overfitting. It is frequently utilized to tasks involving both regression and classification. Random Forest generates a large number of decision trees during training. To reduce overfitting and improve prediction accuracy, each tree is constructed using a random subset of the data and features. In classification tasks, the trees vote with a majority to determine the final prediction. The outcomes for regression problems are combined. Predictions produced by this group decision-making process are reliable and dependable.

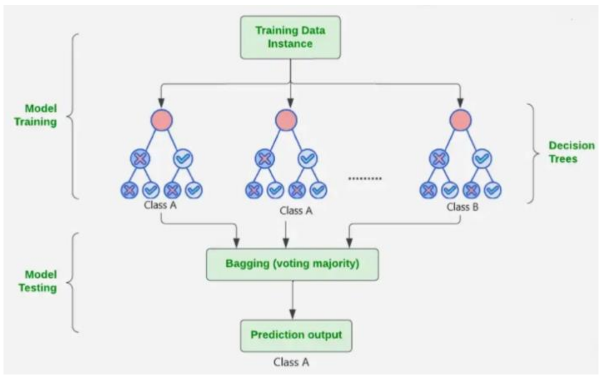


Figure of 2.2.2 RANDOM FOREST

2.2.3 Decision Tree

Decision tree machine learning model is Similar to a flowchart, that makes decisions by splitting data according to characteristic values. Every branch represents a decision rule, and the tree's leaf nodes provide the answer. It's a simple to use and effective tool for regression (numerical value prediction) and classification (classifying data into categories). Decision Trees are frequently utilized for simple prediction tasks because they make it clear how decisions are made based on numerous factors, which makes them easy to understand.

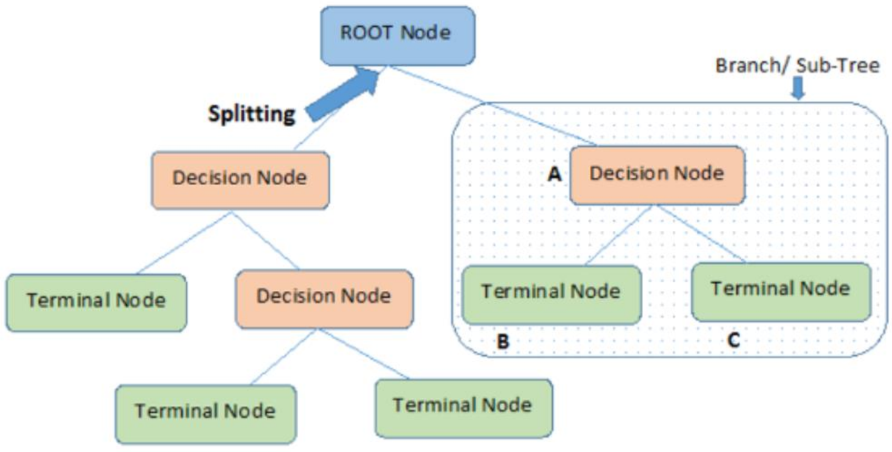
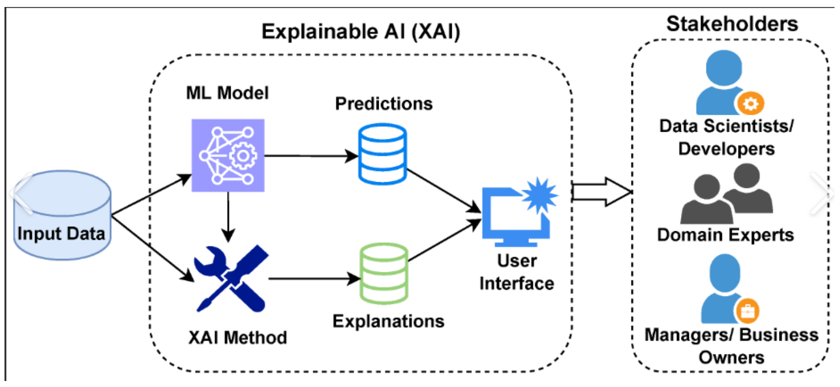


Figure of 2.2.3 Decision Tree.

2.2.4 XAI.

The topic of explainable AI (XAI) aims to improve the human understanding of machine learning algorithms. In comparison with complex models—like traditional ai model as "black boxes" because of their lack of transparency. XAI seeks to clarify the process and reasoning behind a model's specific predictions. More complicated models can be difficult to understand, while simpler models, such as decision trees and linear regression, are easier to interpret since they demonstrate the obvious connects between features and outcomes. XAI helps clients to trust, understand, and validate AI judgments through using techniques like LIME, SHAP, and EL15 to deliver insights into these complicated models. In important field where understanding the reasoning behind predictions is crucial transparency is especially important.



Work flow Figure of 2.2.4 XAI.

CHAPTER-3

SYSTEM ANALYSIS

3.1 OBJECTIVE

The objective of this project is to use machine learning algorithms, such as logistic regression, random forest, and decision trees, to develop a reliable system for identifying credit card fraud. Fraud risk can be better understood with the use of logistic regression, and accuracy can be increased by combining numerous decision trees with random forests. Clear insights into the decision-making process can be obtained from decision trees. Furthermore, Explainable AI (XAI) will be employed to guarantee that the models' predictions are understandable and clear. The ultimate objective is to create an efficient system that achieves a balance between clarity and accuracy.

3.2 ADVANTAGES

- Simple and interpretable; provides probability estimates for fraud.
- Handles large datasets well; reduces overfitting by combining multiple decision trees.
- Easy to understand and visualize; makes decisions based on feature importance.
- Enhances transparency; helps explain model predictions, improving trust and compliance.

3.3 REQUIREMENT ANALYSIS

The act of recognizing, documenting, and monitoring a system's or project's requirements and demands is known as requirement analysis. It entails figuring out what the system's stakeholders believe from it and making sure the finished result lives up to their expectations.

Since it establishes the framework for design, development, and testing, this stage of the project's lifecycle is important.

Steps involved in Requirement Analysis are:

1. **Data Requirements:**
   Large amounts of transaction data, including both genuine and fraudulent transactions, are needed by the system in order to train and evaluate machine learning models.making sure the data is free of prejudices, clean, and consistent, as these could have an impact on the model's performance.
2. **Categorizing Requirements:**
   2.1Functional Requirements

1)Data Collection Module

Gather transaction data from financial institutions or public datasets (Kaggle Credit Card Fraud Detection dataset).Include attributes such as transaction amount, time, and anonymized features.

2) Data Preprocessing Module

Data Cleaning and Normalization:

- Handle missing values by imputation or removal.
- Remove duplicates to ensure data quality.
- Normalize features using techniques such as Min-Max scaling or Standardization.

- Class Imbalance Handling:
    - Apply techniques like under sampling or oversampling to balance

3)Exploratory Data Analysis (EDA) Module

- Descriptive Statistics:
    - Calculate and display summary statistics for the dataset.
- Data Visualization:
    - Use histograms, box plots, pair plots, and heatmaps to understand feature distributions and correlations.

4)Feature Engineering Module

- Feature Selection:
    - Identify and select relevant features for fraud detection.
- Feature Creation:
    - Develop new features if necessary to enhance model performance.

5)Model Development Module

- Machine Learning Algorithms:
    - Implement models such as Logistic Regression, Decision Trees, Random Forest

6) Explainability

- In order to ensure that the choices made by machine learning models are visible and understandable, the system must include Explainable AI techniques. It could be necessary to use methods like LIME or SHAP.

6)Result Visualization Module:

- Visualize transaction data and detection results using dashboards.
- Monitoring:
    - Continuously monitor model performance and update it as needed.

7)Documentation and Reporting Module

- Project Documentation:
    - Document the entire process, including data preprocessing, model development, evaluation, and deployment.

8)Results Presentation:

- Prepare presentations to showcase the project's results and insights. These components ensure that the credit card fraud detection system is comprehensive, accurate, and effective in identifying fraudulent activities.

2.2 Non-Functional Requirements:

1) Performance:

For the system to ensure real-time fraud detection, transactions must be processed with a minimum delaying. This should expand as the number of transactions rises and effectively manage a large volume of transactions.

2) Scalability:
The system needs to be flexible in order to handle increasing data volumes and transaction volumes without experiencing performance loss.

3) Reliability and Availability:
To guarantee constant monitoring and the identification of fraudulent transactions, the system must have a high degree of dependability and minimal disruption.

4) Usability:
Users of different levels of technical skills should be able to easily use the system's interface. Even stakeholders who are not technical should be able to understand the clear, useful information and explanations it provides.

5) Maintainability:
Flexible code that makes improvements and upgrades easy to carry out should make the system maintainable. To help future developers understand and adjust the system as needed, documentation should be thorough.

3.4Modules Description

3.4.1 The Description of Dataset

The Credit Card Fraud Detection dataset comprises 1,048,575 transactions with 23 numerical features and is accessible on Kaggle under the username kalyankaparaju01. Transdate_trans_time, which displays the date and time durations between transactions, the "amt" is amount, which indicates transaction amounts, and “is_fraud”, which indicates fraud labels where 1 for fraudulent and 0 for non-fraudulent, are important features. 6,006 of these transactions are fraudulent and remaining are legit transaction this show dataset is incredibly unbalanced. Preprocessing the data is essential because the overall ratio of categories influences the accuracy and precision of the model.

3.4.2 Dataset Preprocessing

The dataset contained 23 columns at first. Whether the transactions were fraudulent or not was mentioned in the last column. This final column is what we referred to as the target variable or label. Also, the dataset was divided into two groups, X and Y, representing "Features" and "Label," accordingly. We next separated the entire dataset into testing and training sets. Eighty percent of the data is in the training set, while twenty percent is in the testing set.

3.4.3 Data Splitting

The technique of split the available information into two parts for cross-validation is known as "data splitting." The following is the name of the data splits.

- Training Data:

The model that predicts has been trained or prepared for use in making predictions using this part of the data. The machine learning algorithm receives this as input.

- Data for testing:

The model for prediction that was created during the training phase gets evaluated using this part of the data. This is provided to the machine learning model as input in order to determine the model's accuracy.

3.4.4 **Model Development and Training**

Choose appropriate ML algorithms based on the problem, such as Logistic Regression, Decision Trees, Random Forest, Gradient Boosting Machines, Support Vector Machines, Neural Networks, etc. Train the selected models using labelled datasets.

3.4.5 **Model Evaluation and Validation**

Assess model performance using metrics such as accuracy, precision, recall, F1-score, ROC-AUC, and confusion matrix.

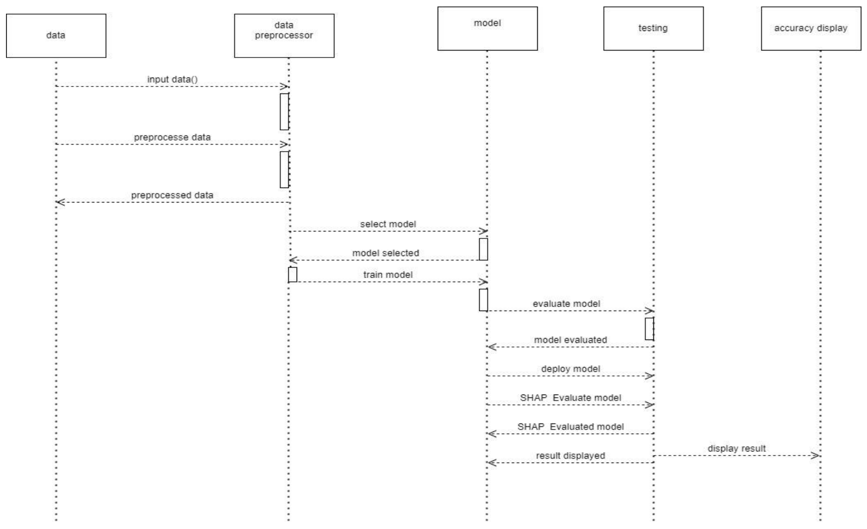
3.4.6 **XAI**

Implement techniques to make model predictions interpretable and understandable Methods like SHAP and LIME to explain the contribution of features to the model's predictions. Visualize decision boundaries, feature importances, and model behaviour.

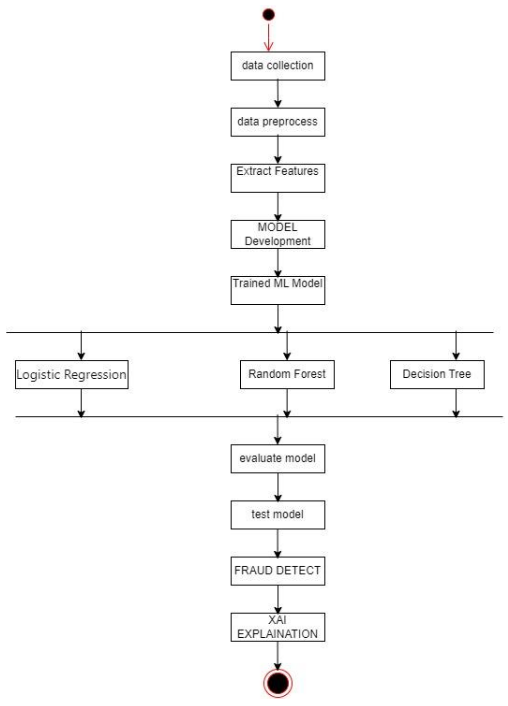
CHAPTER-4

SYSTEM DESIGN

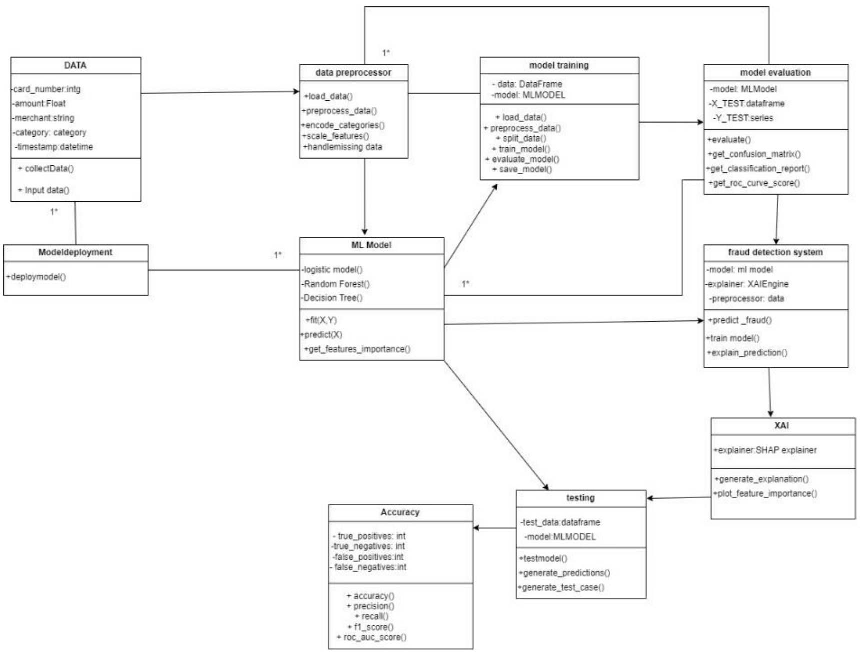
3.1 SEQUENCE DIAGRAM



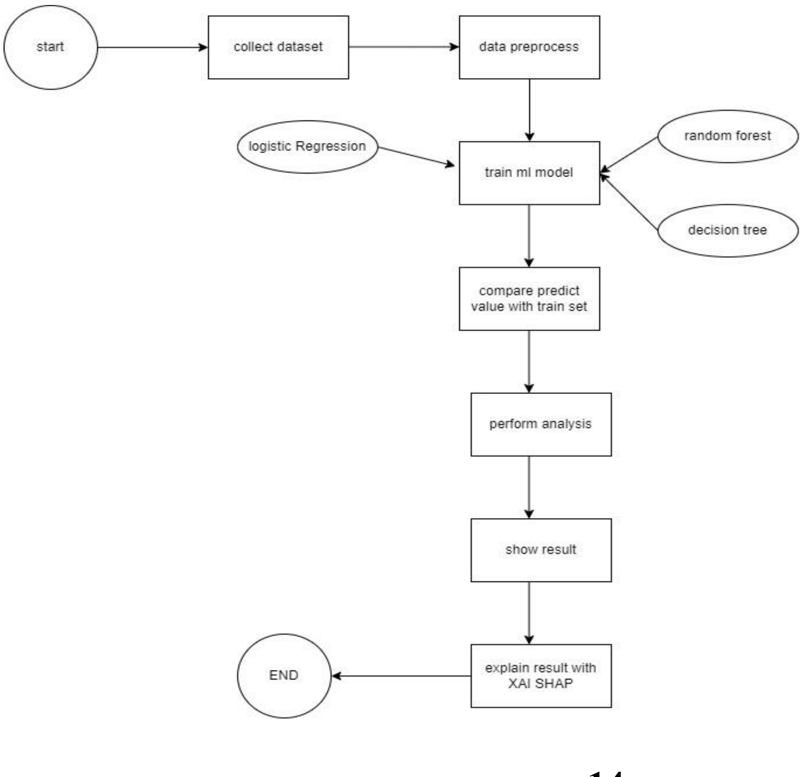
3.2ACTIVITY DIAGRAM



3.3 CLASS DIAGRAM



3.4 METHODLGY DIAGRAM



CHAPTER-5

EXPERIMENTATION

5.1 THE METRICS OF DATA ANALYSIS

The performance, accuracy, and efficiency of data analysis procedures and models are evaluated using quantitative metrics, which are referred to as data analysis metrics. The performance of a model or algorithm in terms of forecasting, spotting trends, or drawing conclusions from the data can be assessed with the use of these indicators. These criteria are essential for evaluating the efficacy of the detection system in the context of credit card fraud detection

5.2 CONFUSION MATRIX

A confusion matrix is a table that displays the actual versus anticipated classifications, thus summarising the performance of a classification model.

- True Positive (TP): Instances where the model correctly predicts the positive class.
- False Positive (FP): Instances where the model incorrectly predicts the positive class.
- False Negative (FN): Instances where the model incorrectly predicts the negative class.
- True Negative (TN): Instances where the model correctly predicts the negative class.

5.3 PRECISION, RECALL, ACCURACY & FL-SCORE

- **Precision:** Precision measures the accuracy of positive predictions. It tells you how many of the items identified as positive are actually positive.

$$Precision = \frac{\mathrm{TP}}{TP + FP}$$

- **Accuracy**: Accuracy is defined as the percentage of true positives and true negatives among all instances that were accurately predicted. It shows the proportion of accurate predictions the model produces.

$$Accuracy = \frac{TP + TN}{\mathrm{FP} + \mathrm{FN} + \mathrm{TP} + \mathrm{TN}}$$

.

- **Recall:** (True Positive Rate or Sensitivity)
  The ratio of accurately anticipated positive cases, or true positives, to the total number of actual positive instances, or true positives + false negatives, is known as recall, or sensitivity. It assesses how well the model can account for all pertinent positive examples.

$$Recall = \frac{\mathrm{TP}}{TP + TN}$$

F1 Score: The F1 Score is the harmonic mean of recall and precision. It offers a solitary metric that harmonises the model's recall and precision.

$$F1Score = 2 * \frac{\mathrm{Precision} * \mathrm{Recall}}{2aPrecision + Recall}$$

CHAPTER 6

RESULT ANALYSIS DISCUSSION

6.1 DATASET

The Credit Card Fraud Detection dataset comprises 1,048,575 transactions with 23 numerical features and is accessible on Kaggle and 6,006 of these transactions are fraudulent and remaining are legit transaction. Preprocessing the data is essential because the overall ratio of categories influences the accuracy and precision of the model.

Category	**Values**
Total transactions	1042569
Non-Fraud Transactions	6006
Train set	838860
Test set	209715

Table 6.1.1: Description of the dataset

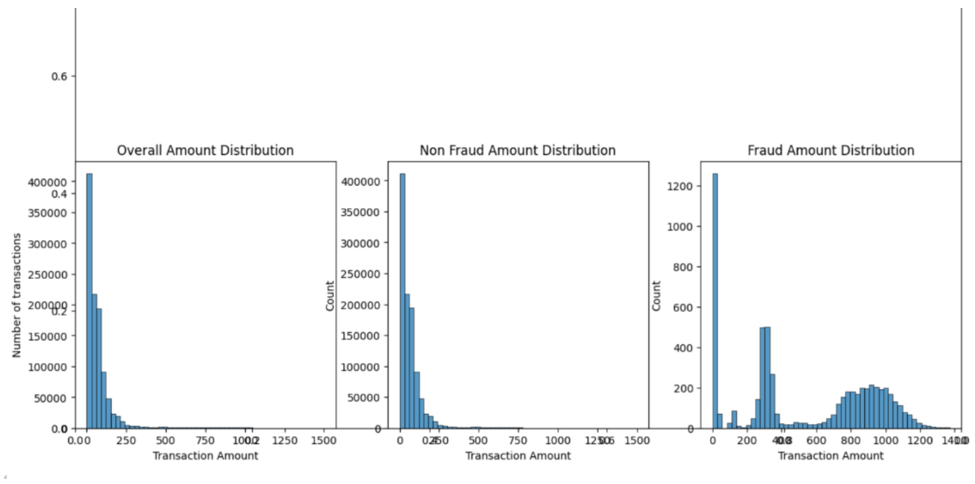


Figure 6.1.2 graph of dataset

The overall distribution of transaction values for whole, fraudulent, and non-fraudulent transactions is shown in the graph.

- **Distribution of Total amounts:** Most transactions involve small numbers, and when transaction numbers increase, they drop off significantly.
- **The distribution of non-fraud amounts:** the sample is consistent with the overall distribution, indicating that the majority of transactions are not fraudulent and typically contain lesser amounts.
- **Fraud Amount Distribution**: There is a noticeable increase at higher amounts (about 600–1200), but the fraudulent transactions are more widely dispersed over a larger range of quantities. This implies that although there are quite a few of low-amount fraudulent transactions, fraud is more likely to happen in larger transactions.

6.2 DATASET BALANCE:

To balance dataset, we used "fnew = legit.sample(n=6006)" is used in the context of data sampling in Python, typically when working with a pandas DataFrame or Series. This is often used in data preprocessing to either reduce the size of the dataset or to balance the dataset, for example, by creating a balanced dataset where the number of legitimate transactions is equal to the number of fraudulent ones.

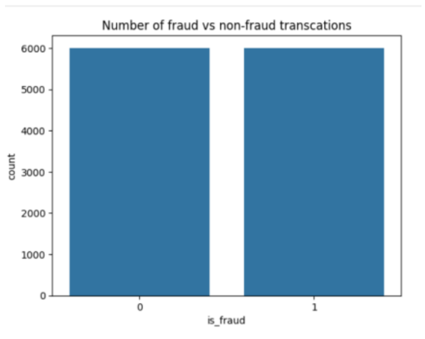


figure of 6.2 balanced dataset

6.2.1 performance Metrics

We now divided the balanced dataset into "train" and "test" sets in order to run trials of our models. Here, we modified our dataset to create a train set and test set that are both balanced. Since training data is necessary for guided machine learning models, we choose to train 8,38,860 transactions and test 2,09,715transactions on the model.

We used accuracy, precision, recall, and Fl-Score, Sensitivity, Specificity metrics to assess the models' overall performance across the dataset. The balanced dataset's results are displayed in fig 6.2. and displays the performance metrics for the balanced dataset model simulations.

Model Name	Accuracy	F1 Score	Precision	Recall	ROC
Logistic Regression	0.846858	0.835125	0.747994	0.837946	0.945233
Random Forest	0.927591	0.930177	0.930177	0.978575	0.930177
Decision Tree	0.926342	0.929172	0.931782	0.971887	0.926576

Table of 6.2.1 performance Metrics

This Table 6.2.1 shows a performance matrix comparing different machine learning models on a balanced dataset. The models evaluated are Logistic Regression, Random Forest, and Decision Tree. The metrics used for comparison are Accuracy, F1 Score, Precision, Recall, and ROC.

Logistic Regression:

- Has the lowest accuracy (0.846858) and F1 Score (0.835125) among the three models.
- Precision (0.747994) and Recall (0.837946) are relatively lower, indicating some trade-offs in predicting the positive class.
- However, it has the highest ROC AUC (0.945233), suggesting good overall model discrimination.

Random Forest:

- Shows the highest accuracy (0.927591) and F1 Score (0.930177), indicating strong overall performance.
- Precision and Recall are both equal at 0.930177 and 0.978575, respectively, showing a good balance between false positives and false negatives.
- The ROC AUC (0.930177) is slightly lower than that of Logistic Regression but still indicates strong performance.

Decision Tree:

- Has slightly lower accuracy (0.926342) and F1 Score (0.929172) than the Random Forest, but still performs well.
- Precision (0.931782) is the highest among all models, indicating fewer false positives.
- Recall (0.971887) is high, similar to Random Forest, showing effective detection of true positives.
- The ROC AUC (0.926576) is the lowest among the three models, but it is still quite strong.

The Random Forest model stands out with the best overall performance across most metrics, followed closely by the Decision Tre

6.3 Classification report

LOGISTIC REGRESSION:

	precision	recall	F1 score	support
False	0.78	0.95	0.86	1157
True	0.95	0.75	0.84	1246
accuracy			0.85	2403
Macro average	0.86	0.85	0.85	2403
Weighted average	0.86	0.85	0.85	2403

Table of 6.3.1 Classification report for logistic regression

The Logistic Regression model achieves an overall accuracy of 0.85, meaning 85% of predictions are correct. It has a precision of 0.78 for the "False" class and 0.95 for the "True" class, indicating the model is better at correctly identifying the "True" class. The recall for the "False" class is high at 0.95, but lower for the "True" class at 0.75, meaning the model is more likely to miss true positives. The F1 scores are similar for both classes, around 0.84 to 0.86, reflecting a balanced trade-off between precision and recall. The model performs consistently across these metrics, with both macro and weighted averages at 0.85, indicating stable and reliable classification performance across the dataset.

Random Forest:

	precision	recall	F1 score	support
False	0.92	0.92	0.92	1157
True	0.93	0.93	0.93	1246
accuracy			0.93	2403
Macro average	0.93	0.93	0.93	2403
Weighted average	0.93	0.93	0.93	2403

Table of 6.3.2 Classification report for Random Forest

The Random Forest model shows excellent overall performance, with an accuracy of 0.93, meaning 93% of its predictions are correct. Both classes ("False" and "True") have very similar precision, recall, and F1 scores, all around 0.92 to 0.93. This indicates that the model is highly effective at correctly identifying both classes, with a balanced approach to minimizing both false positives and false negatives. The macro and weighted averages are also consistently at 0.93, suggesting that the model performs reliably across the dataset, providing strong and stable classification for both class

Decision Tree:

	precision	recall	F1 score	support
False	0.93	0.92	0.92	1157
True	0.93	0.93	0.93	1246
accuracy			0.93	2403
Macro average	0.93	0.93	0.93	2403
Weighted average	1.93	0.93	0.93	2403

Table of 6.3.3 Classification report for Decision Tree

This table presents a classification report for a Decision Tree model. The model demonstrates strong performance across various metrics. It has an overall accuracy of 93%, with consistent precision and recall scores of 0.93 for both the True and False classes. The F1 scores, which balance precision and recall, are also high at 0.92 for the False class and 0.93 for the True class. The support column shows a relatively balanced dataset with 1157 instances in the False class and 1246 in the True class, totaling 2403 samples. The macro average, which gives equal weight to each class, matches the overall accuracy at 0.93 across precision, recall, and F1 score. The weighted average shows a slightly inflated precision of 1.93, which might be a typographical error and should likely be 0.93 to align with the other metrics. Overall, this Decision Tree classifier appears to be performing very well, with balanced and high performance across both classes.

6.4 More Specific classification report:

Model	Accuracy	sensitivity	Specificity	F1 score
Logistic Regression	0.8468580	0.7479935	0.9533275	0.8351254
Random Forest	0.9275905	0.9301765	0.9248055	0.9301765
Decision Tree	0.9263420	0.931781	0.9204840	0.9291716

Table of 6.4 More Specific Classification report

This table compares the performance of three different classification models: Logistic Regression, Random Forest, and Decision Tree. The Random Forest model shows the best overall performance, with the highest accuracy (0.9275905), sensitivity (0.9301765), and F1 score (0.9301765). It also has a high specificity (0.9248055). The Decision Tree model performs similarly well, with slightly lower but still impressive metrics across the board. Logistic Regression, while still performing decently, lags behind the other two models in all metrics, particularly in sensitivity (0.7479935). This suggests that tree-based models (Random Forest and Decision Tree) are more effective for this particular classification task, with Random Forest having a slight edge. The high specificity of Logistic Regression (0.9533275) indicates it's good at identifying true negatives, but its lower sensitivity suggests it might miss some positive cases compared to the other models.

6.5 ROC CURVE ANALYSIS

One of the most important tools for assessing how well binary classification algorithms work is ROC curve analysis. The term "Receiver Operating Characteristic" (ROC) refers to a graphical depiction that shows how different threshold settings affect the trade-off between the genuine positive rate (sensitivity) and the false positive rate (1-specificity).

By plotting these rates on the y- and x-axes, respectively, the ROC curve illustrates a model's diagnostic capability. The model's overall performance is quantified by the area under the ROC curve (AUC), where a value of 1 represents perfect classification and a value of 0.5 indicates no discriminative power beyond random chance. ROC curve analysis is widely used in fields like machine learning and medical diagnostics to evaluate the effectiveness of predictive algorithms, determine the optimal classification threshold, and compare the performance of different models.

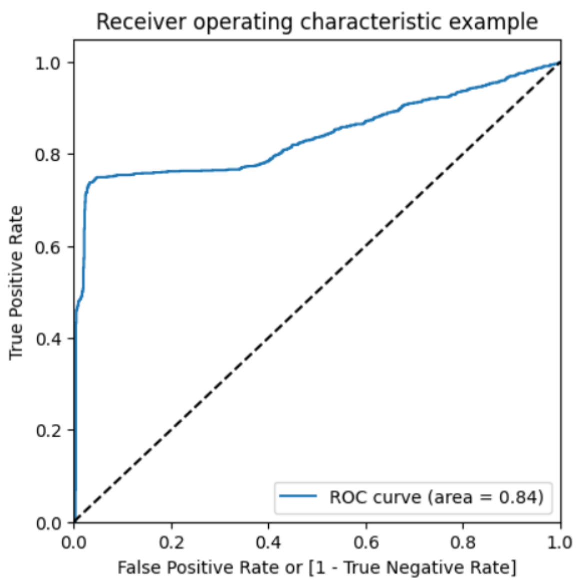


Figure 6.5.1 logistic regression

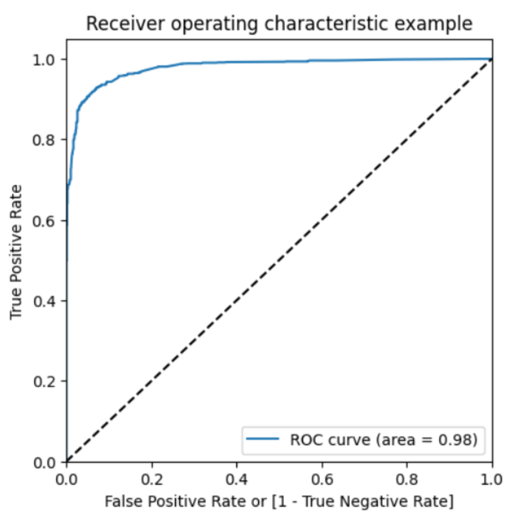


Figure 6.5.2 of random forest

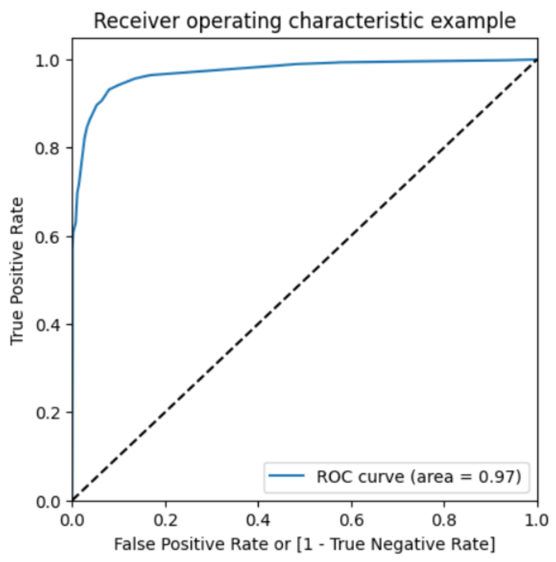


Figure 6.5.3 of decision tree

Based on the ROC (Receiver Operating Characteristic) curves shown in the three images we can say:

1. Logistic Regression (Figure 6.5.1):

The ROC curve has an area under the curve (AUC) of 0.84. This indicates good but not excellent performance. The curve rises sharply at first, then levels off, suggesting the model has good discrimination ability, especially at lower false positive rates.

2. Random Forest (Figure 6.5.2):

This model shows excellent performance with an AUC of 0.98. The curve rises very steeply and quickly approaches the top-left corner, indicating high true positive rates even at very low false positive rates. This suggests superior discriminative power compared to logistic regression.

3. Decision Tree (Figure 6.5.3):

The decision tree model also performs very well, with an AUC of 0.97. Its ROC curve is very similar to that of the random forest, showing steep initial rise and overall excellent performance.

We can say all three models perform well, but the tree-based models (random forest and decision tree) outperform logistic regression. The random forest model shows slightly better performance than the decision tree, but both are excellent. This comparison suggests that for this particular classification task, tree-based models are more effective at distinguishing between classes across various threshold settings.

6.6 Confusion Matrix Analysis

The confusion matrix for Logistic Regression, Random Forest, Decision Tree models**.**

LOGISTIC REGRESSION:

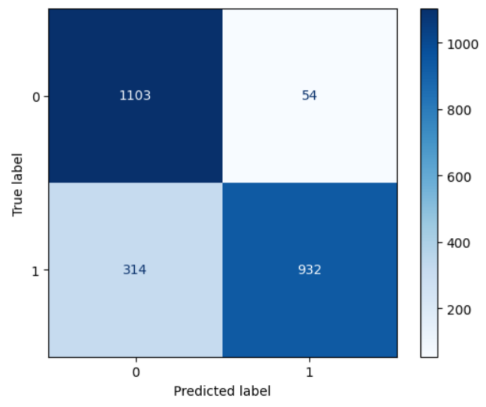


Figure 6.6.1 of logistic regression

The confusion matrix shown in the image represents the performance of a Logistic Regression model on a balanced dataset for credit card fraud detection. The matrix provides a breakdown of the model's predictions:

- **True Negatives (TN):** 1103 These are legitimate transactions correctly identified as not fraudulent.
- **False Positives (FP):** 54 These are legitimate transactions incorrectly flagged as fraudulent.
- **False Negatives (FN):** 314 These are fraudulent transactions incorrectly classified as legitimate.
- **True Positives (TP):** 932 These are fraudulent transactions correctly identified as fraudulent.

The model tends to miss a relatively high number of fraudulent transactions (314 FN), indicating a risk of failing to detect fraud. However, it does a good job of identifying legitimate transactions, with only 54 legitimate transactions being falsely flagged as fraudulent.

RANDOM FOREST:

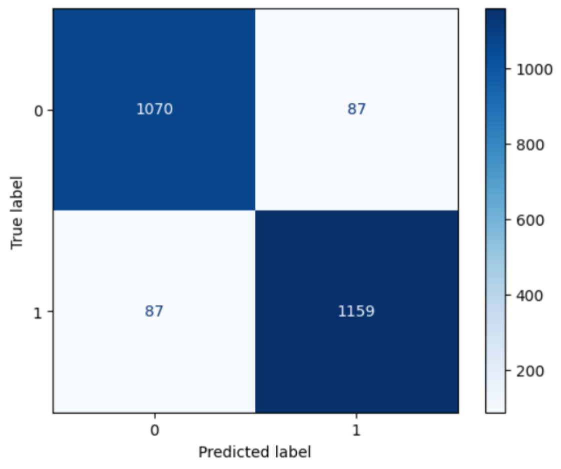


Figure 6.6.2 of Random Forest

The confusion matrix shown in the image represents the performance of a Random Forest model on a balanced dataset for credit card fraud detection. The matrix provides a breakdown of the model's predictions:

- **True Negatives (TN)**: 1070 Legitimate transactions correctly identified as not fraudulent.
- **False Positives (FP)**: 87 Legitimate transactions incorrectly flagged as fraudulent.
- **False Negatives (FN)**: 87 Fraudulent transactions incorrectly classified as legitimate.
- **True Positives (TP)**: 1159 Fraudulent transactions correctly identified as fraudulent.

This model performs well in identifying fraudulent transactions, with only 87 cases missed (FN). However, it has a slightly higher rate of falsely identifying legitimate transactions as fraudulent compared to the Logistic Regression model.

DECISION TREE:

The confusion matrix shown in the image represents the performance of a Decision Tree model on a balanced dataset for credit card fraud detection. The matrix provides a breakdown of the model’s predictions:

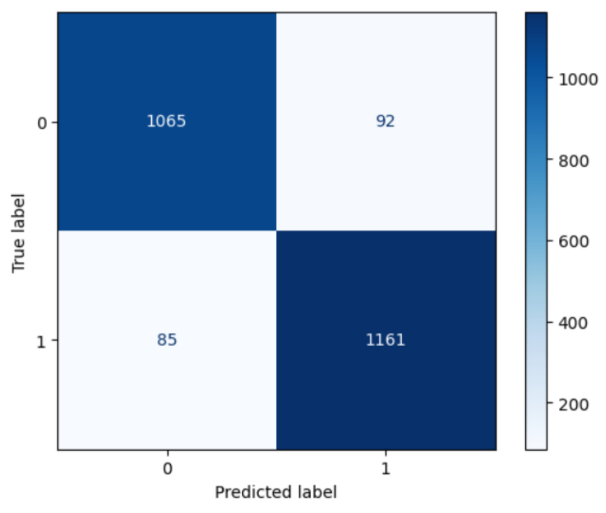


Figure 6.6.2 of Decision Tree

- **True Negatives (TN):** 1065 Legitimate transactions correctly identified as not fraudulent.
- **False Positives (FP):** 92 Legitimate transactions incorrectly flagged as fraudulent.
- **False Negatives (FN):** 85 Fraudulent transactions incorrectly classified as legitimate.
- **True Positives (TP):** 1161 Fraudulent transactions correctly identified as fraudulent.

The Decision Tree model has the best performance in terms of detecting fraudulent transactions, with only 85 fraudulent cases missed. It also has a relatively low number of false positives, meaning it does not unnecessarily flag too many legitimate transactions as fraudulent.

In summary, The Decision Tree model has the highest True Positive rate (1161), indicating it is better at correctly identifying positive cases compared to the other models. The Random Forest model has a similar performance to the Decision Tree but with slightly fewer false negatives and false positives.

The Logistic Regression model has a higher number of false negatives (314), indicating it misses more positive cases compared to the other models**.**

6.7 XAI :

6.7.1 SHAP:

A framework called SHAP is used to analyse machine learning model output. It provides a way of understanding how each input feature affects the predictions made by the model. SHAP improves our understanding of machine learning model operation. We can clearly see how each feature affects predictions thanks to SHAP. This makes things easier for everyone to understand while also ensuring fairness.
Because it shows the importance of every characteristic in prediction, SHAP is helpful. Shapley values are useful in understanding complex models and the effect of input features on predictions.

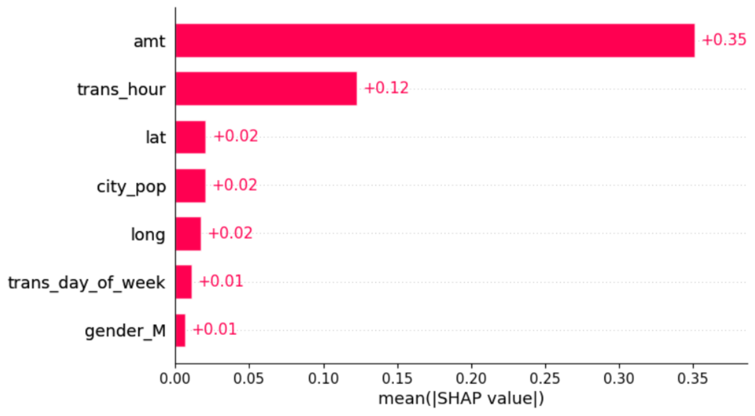


Figure 6.7.1 Shap bar plot

The image, which is a SHAP summary plot, displays the importance of various attributes in a machine learning model that is probably going to be utilized in predicting credit card fraud.

The mean of the SHAP values, which show the average impact of each feature on the model's output, are represented by the x-axis. The more impact a feature has on the model's predictions, the higher its SHAP score.

The analysis shows that the transaction amount (amt) is the most significant driver of the model's predictions, with a mean SHAP value of +0.35, indicating that this feature has the highest average impact on the prediction outcomes. The transaction hour (trans_hour) is the second most influential feature, with a mean SHAP value of +0.12, suggesting that the time of the transaction plays a considerable role in the model’s decision-making process.

Other features, including latitude (lat), city population (city_pop), longitude (long), transaction day of the week (trans_day_of_week), and gender (gender_M), exhibit much lower mean SHAP values, each around +0.01 to +0.02. These features have a relatively minor influence on the model's predictions compared to amt and trans_hour.

The graph highlights how much the model depends on the number and timing of transactions, with little impact from geographical, demographics, and time factors such as gender, day of the week, and location coordinates. This understanding is essential for comprehending the model's decision-making process, particularly in an area like fraud detection where stakeholders' decision-making can be improved and trust can be increased by being able to explain predictions.

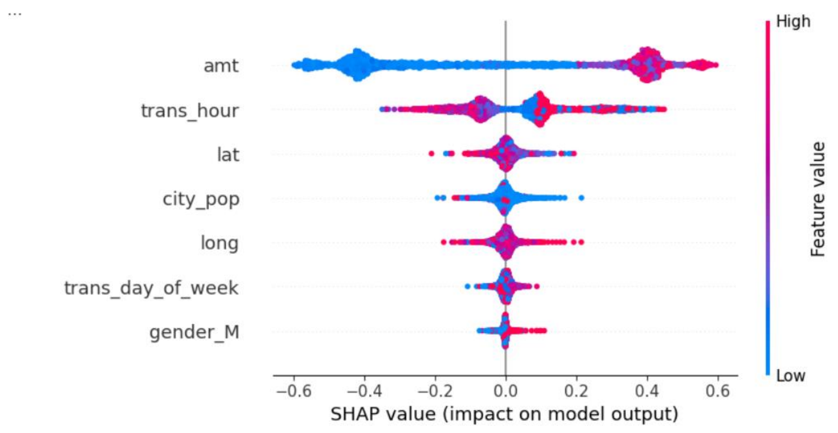


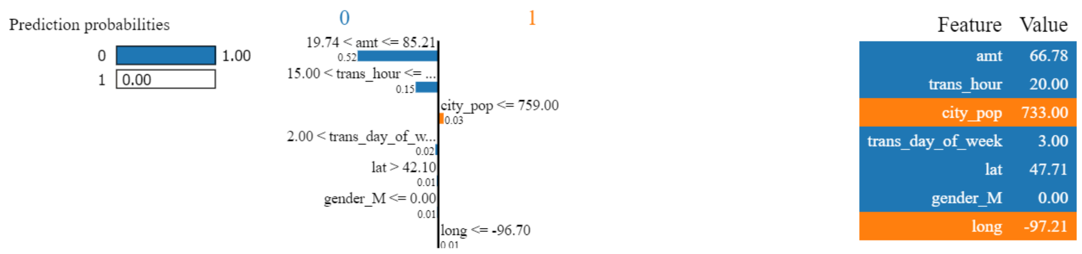
Figure 6.7.1.2 Shap summary plot

The graph shows the SHAP values for each feature across all of the dataset's samples and is a SHAP summary plot from an alternate perspective. A clearer picture of how each feature affects the model's predictions is provided by this figure 6.7.2. X-axis in The SHAP value shows the amount that a feature modifies the prediction, either positively or negatively. The SHAP summary plot shows that the transaction amount (amt) is the most influential feature, with a wide range of SHAP values, indicating it strongly impacts the model's predictions. High transaction amounts generally increase the predicted probability, while lower amounts decrease it. The transaction hour (trans_hour) also plays a significant role, with mixed effects depending on the hour. Other features like latitude, longitude, city population, day of the week, and gender have a much smaller impact, with SHAP values clustered around zero, indicating they have minimal influence on the model's output. This analysis highlights that the model primarily relies on transaction amount and timing in its decision-making process.

6.7.2 LIME:

Explainable Artificial Intelligence (XAI) uses a technique called LIME (Local Interpretable Model-agnostic Explanations) to make complicated, black-box machine learning models interpretable. It functions by using a more straightforward, interpretable model (such a decision tree or linear model) to approximate the model narrowly around a particular prediction. Users can obtain insights into how the more advanced model behaves in a given situation by understanding the decision-making process of this smaller model.

LIME is a flexible tool for explaining specific predictions since it is algorithm-agnostic, meaning it may be used with any kind of machine learning model.



Feature	Value
amt	66.78
trans_hour	20.00
city_pop	733.00
trans_day_of_week	3.00
lat	47.71
gender_M	0.00
long	-97.21

Figure of 6.8 LIME

The LIME graph provides an explanation for a specific prediction made by a machine learning model, likely related to credit card fraud detection. The left side shows the predicted probabilities for each class, with "0" representing non-fraudulent transactions and "1" representing fraudulent transactions. In this case, the model predicts a probability of 1.00 (or 100%) for class "0," indicating high confidence that the transaction is not fraudulent.

The middle section visualizes the contribution of various features towards this prediction. Features pushing the prediction towards "0" are shown on the left in blue, while features pushing towards "1" would be shown on the right in orange (though in this case, none are pushing towards "1").

For instance, the feature "amt" (amount) in the range 19.74 < amt <= 85.21 contributes significantly towards predicting a non-fraudulent transaction, as does "trans_hour" (transaction hour) being less than or equal to 15.00. On the right side, the actual feature values for this instance are listed.

These include the transaction amount (66.78), transaction hour (20:00), city population (733.00), transaction day of the week (3), latitude (47.71), gender (Male, coded as 0), and longitude (-97.21). These feature values help contextualize the specific transaction being analyzed and how the model interprets these values.

CHAPTER 7

CONCLUSION AND FUTURE WORK

Credit card fraud is an increasing issue, with fraudsters constantly developing new tactics. This project explores recent advances in fraud detection, focusing on machine learning (ML) and explainable artificial intelligence (XAI) methods. The goal is to accurately detect fraud while minimizing false positives. The effectiveness of ML and XAI depends largely on the quality of input data and the features used. In this study, Random Forest showed the highest accuracy (99.63%) among the tested models. Although under sampling helped address data imbalance, the process was time-consuming. While current performance is strong, ongoing efforts aim to reduce false negatives and improve accuracy by combining multiple algorithms and incorporating more real-world data.

The future of credit card fraud detection using machine learning (ML) and explainable AI (XAI) is promising, with advancements likely to focus on improving real-time detection accuracy and reducing false positives and negatives. As fraud tactics evolve, ML models will increasingly rely on adaptive algorithms capable of learning from new patterns and anomalies in vast and dynamic datasets. XAI will play a crucial role in making these models more transparent and interpretable, helping businesses and regulators trust and fine-tune their fraud detection systems. Additionally, integrating multiple algorithms and leveraging deep learning could further enhance detection capabilities, making fraud detection more robust and scalable in an ever-changing landscape.

CHAPTER 8

CODE

- CELL-1

```
!pip install -q kaggle
!mkdir ~/.kaggle
ls~/.kaggle
```

- cell-2 (dataset download)

```
!kaggle datasets download -d kalyankaparaju01/credit-card-fraud-
detection-dataset
```

- CELL-3

```
import zipfile
with zipfile.ZipFile("credit-card-fraud-detection-dataset.zip","r") as file:
    file.extractall()
```

- CELL -4

```
%pip install seaborn
%pip install matplotlib
```

- CELL-5

```
# Exploratory Data Analysis (EDA) Modules
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

- CELL-6

```
# Data Cleaning and Analysis Modules
from sklearn.model_selection import StratifiedKFold
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV
```

- CELL-7

```
# Model Evaluation Modules
from sklearn.model_selection import cross_val_score
from sklearn.metrics import precision_score, accuracy_score, f1_score,
r2_score
```

```
from sklearn.metrics import precision_recall_curve, roc_curve
from sklearn.metrics import recall_score
```

- CELL-8

```
df = pd.read_csv('fraudTrain.csv')
df.head()
```

- CELL-9

```
len(df)
#dtypes of the columns
df.dtypes
```

- CELL-10

```
#converting trans_date_trans_time into datetime
df['trans_date_trans_time'] = pd.to_datetime(df['trans_date_trans_time'])
print(df.dtypes['trans_date_trans_time'])
```

- CELL-11

```
# deriving additonal columns from 'trans_date_trans_time'
#deriving hour
df['trans_hour'] = df['trans_date_trans_time'].dt.hour
#deriving 'day of the week'
df['trans_day_of_week'] = df['trans_date_trans_time'].dt.dayofweek + 1
df['trans_day_of_week'] = df['trans_day_of_week'].astype(int)
#deriving 'year_month'
df['trans_year_month'] = df['trans_date_trans_time'].dt.to_period('M')
```

- CELL-12

```
#converting data types that should be categorical into "category"
df['category'] = df['category'].astype('category')
df['gender'] = df['gender'].astype('category')
df['is_fraud'] = df['is_fraud'].astype('category')
```

- CELL-13

```
df.dtypes
```

- CELL-14

```
#dropping variables which are not usefull for the Visualization and
analytics
df.drop(['trans_date_trans_time','first', 'last', 'dob',] , axis=1,
inplace=True)
```

- CELL-15
  df.info()

- CELL-16
  #let us look at the number of unique values in the dataset
  df.nunique()

- CELL-17
  df.sample()

- CELL-18
  #determing the shape of the dataset
  df.shape

- CELL-19
  df.columns

- CELL-20
  #describing the dataset
  df.describe()

- CELL-21
  # checking the number of missing values in each column
  df.isnull().sum()

- CELL-22
  # distribution of legit transactions & fraudulent transactions
  df['is_fraud'].value_counts()

- CELL-23
  #plotting the above distributions
  fig = plt.subplots(figsize=(15,10))
  plots = []
  #plotting the amt feature
  #distribution plots
  plots.append(sns.histplot(df[df.amt <= 1500].amt, bins=50,
  ax=plt.subplot(234)))
  plots.append(sns.histplot(df[(df.is_fraud==0) & (df.amt<=1500)].amt,
  bins=50, ax=plt.subplot(235)))
  plots.append(sns.histplot(df[(df.is_fraud==1) & (df.amt<=1500)].amt,
bins=50, ax=plt.subplot(236)))

```
#setting titles
plots[0].set_title('Overall Amount Distribution')
plots[1].set_title('Non Fraud Amount Distribution')
plots[2].set_title('Fraud Amount Distribution')

#setting x labels
plots[0].set_xlabel('Transaction Amount')
plots[1].set_xlabel('Transaction Amount')
plots[2].set_xlabel('Transaction Amount')

#setting y label
plots[0].set_ylabel('Number of transactions')

plt.show()
```

- CELL-24

```
#year_month vs number of transactions
df_timeline01 =
df.groupby(df['trans_year_month'])[['trans_num','cc_num']].nunique().res
et_index()
df_timeline01.columns =
['year_month','num_of_transactions','customers']
df_timeline01
```

- CELL-25

```
# Set up the plot for is_fraud = 1
fig, ax1 = plt.subplots(figsize=(14, 8))

# Get the order of states by their count in descending order
state_order = df[df['is_fraud'] == 1]['state'].value_counts().index

# Plotting the count of each state for is_fraud = 1
plot1 = sns.countplot(x='state', data=df[df['is_fraud'] == 1],
order=state_order, palette='Set2', ax=ax1)
plot1.set_xticklabels(plot1.get_xticklabels(), rotation=90)

ax1.set_title('Transaction Count by State in a fraudulent transaction
(Descending Order)')

# Show the plot for is_fraud = 1
plt.show()
```

- CELL-26

```
# Calculate fraud ratios for all states
fraud_ratios = []

for state in df['state'].unique():
    total_count = df[df['state'] == state].shape[0]
    fraud_count = df[(df['state'] == state) & (df['is_fraud'] == 1)].shape[0]
    ratio = fraud_count / total_count if total_count > 0 else 0
    fraud_ratios.append({'State': state, 'Fraud Ratio': ratio, 'Total 
Transactions': total_count})

# Create a DataFrame from the list of dictionaries
fraud_ratio_df = pd.DataFrame(fraud_ratios)

# Print the DataFrame
print(fraud_ratio_df)
```

- CELL-27

```
sns.countplot(x='is_fraud', data = df)
plt.title('Number of fraud vs non-fraud transcations')
plt.show()
```

- CELL-28

```
# separating the data for analysis
legit = df[df.is_fraud == 0]
fraud = df[df.is_fraud == 1]
```

- CELL-29

```
   print(legit.shape)
   print(fraud.shape)
```

- CELL-30

```
fraud['is_fraud'].value_counts()
legit['is_fraud'].value_counts()
```

- CELL-31

```
fnew = legit.sample(n=6006)
d_new = pd.concat([fnew, fraud], axis=0)
d_new['is_fraud'].value_counts()
```

- CELL-32
```
d_new.shape
```

- CELL-33
```
sns.countplot(x='is_fraud', data = d_new)
plt.title('Number of fraud vs non-fraud transcations')
plt.show()
```

- CELL-34
```
# Import library
from sklearn.model_selection import train_test_split
X =  [ 'amt', 'gender','lat', 'long', 'city_pop',
'trans_hour','trans_day_of_week','is_fraud']
df= pd.get_dummies(df[X], drop_first=True)
predictors =  [ 'amt',  'city_pop', 'trans_hour', 'gender_M','lat',
'long','trans_day_of_week']
```

- CELL-35
```
# partition data
X = df[predictors]
Y = df[ 'is_fraud_1']
```

- CELL-36
```
# Splitting data into train and test set 80:20
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, train_size=0.8,
test_size=0.2, random_state=100)
```

- CELL-37
```
print(X.shape, X_train.shape, X_test.shape)
```

- CELL-38
```
print(X)
print(Y)
```
- CELL-39
```
# Impoting metrics
from sklearn import metrics
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay,
classification_report, f1_score, precision_score, recall_score
results = pd.DataFrame(columns=['Model Name', 'Accuracy', 'F1-score',
'ROC','precision','recall'])
```

- CELL-40

```
# ROC Curve function
def draw_roc( actual, probs ):
    fpr, tpr, thresholds = metrics.roc_curve( actual, probs,
                                              drop_intermediate = False )
    auc_score = metrics.roc_auc_score( actual, probs )
    plt.figure(figsize=(5, 5))
    plt.plot( fpr, tpr, label='ROC curve (area = %0.2f)' % auc_score )
    plt.plot([0, 1], [0, 1], 'k--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate or [1 - True Negative Rate]')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic example')
    plt.legend(loc="lower right")
    plt.show()
    return None
```

- CELL-41 (LOGISTIC REGRESSION)

```
# Importing scikit logistic regression module
from sklearn.linear_model import LogisticRegression
```

- CELL-42

```
# Instantiate the model with best C
logistic = LogisticRegression(C=0.01)
```

- CELL-43

```
# Fit the model on the train set
logistic_model = logistic.fit(X_train, Y_train)
```

- CELL-44

```
# Prepare results function
def display_test_results(model_name, model):
    # Prediction on the test set
    Y_test_pred = model.predict(X_test)
 # Confusion matrix
    print("----------------- Confusion Matrix --------------------")
    c_matrix = metrics.confusion_matrix(Y_test, Y_test_pred)
    print(c_matrix)
    cm_display = ConfusionMatrixDisplay(confusion_matrix=c_matrix)
    cm_display.plot(cmap=plt.cm.Blues)
    plt.show()
    # classification_report
    print("----------------- classification_report -------------------")
    print(classification_report(Y_test, Y_test_pred))
    print("----------------- More Specific classification_report ----------------
----")
    TP = c_matrix[1,1] # true positive
    TN = c_matrix[0,0] # true negatives
    FP = c_matrix[0,1] # false positives
    FN = c_matrix[1,0] # false negatives

    # Accuracy
    print("Accuracy:-",metrics.accuracy_score(Y_test, Y_test_pred))
    # Sensitivity
    print("Sensitivity:-",TP / float(TP+FN))
    # Specificity
    print("Specificity:-", TN / float(TN+FP))

  # F1 score
    print("F1-Score:-", f1_score(Y_test, Y_test_pred))
    # Predicted probability
    Y_test_pred_proba = model.predict_proba(X_test)[:,1]
# roc_auc
    print("----------------- ROC -------------------")
    roc_auc = metrics.roc_auc_score(Y_test, Y_test_pred_proba)

    # Plot the ROC curve
    draw_roc(Y_test, Y_test_pred_proba)
  # add all metrics score in final result store
    results.loc[len(results)] = [model_name,
metrics.accuracy_score(Y_test, Y_test_pred), f1_score(Y_test,
Y_test_pred), precision_score(Y_test, Y_test_pred),
    recall_score(Y_test, Y_test_pred),roc_auc]
    return None
```

- CELL-45

```
display_test_results("Logistic Regression", logistic_model)
```

- CELL-46 (RANDOM FOREST)

```
# Importing random forest classifier
from sklearn.ensemble import RandomForestClassifier
```

- CELL-47
```
random_forest_model = RandomForestClassifier(bootstrap=True,
                            max_depth=5,
                            min_samples_leaf=50,
                            min_samples_split=50,
                            max_features=10,
                            n_estimators=100)
```

- CELL-48
```
# Fit the model
random_forest_model.fit(X_train, Y_train)
```

- CELL-49
```
display_test_results("Random Forest", random_forest_model)
```

- CELL-50 (DECISION TREE)
```
# Importing decision tree classifier
from sklearn.tree import DecisionTreeClassifier
```

- CELL-51
```
    # Model with optimal hyperparameters
decision_tree_model = DecisionTreeClassifier(criterion = "gini",
                            random_state = 100,
                            max_depth=5,
    min_samples_leaf=100,
    min_samples_split=100)
    decision_tree_model.fit(X_train, Y_train)
```

- CELL-52
```
display_test_results("Decision Tree", decision_tree_model)
```

- CELL-53
```
results.sort_values(by="ROC", ascending=False)
```

- CELL-54 (SHAP)
```
%pip install shap
%pip install xgboost
```
- CELL-55
```
import xgboost as xgb
import shap
shap.initjs()
```

- CELL-56

```
# train xg boost model
model = xgb.XGBRegressor(objective='reg:squarederror', random_state
=0)
model.fit(X_train, Y_train)
```

- CELL-57

```
# explain prediction with shap
explainer = shap.Explainer(model)
shap_values = explainer(X_test)
```

- CELL-58

```
#Mean SHAP
shap.plots.bar(shap_values)
```

- CELL-57

```
#summary plot of shap values
shap.summary_plot(shap_values,X_test)
```

- CELL-58

```
!pip install lime
!pip install --upgrade xgboost lime
!pip install scikit-learn
```

- CELL-59

```
#lime package
import lime
import lime.lime_tabular
```

- CELL-60

```
# Example setup
from sklearn.preprocessing import LabelEncoder

# Assuming df is your original DataFrame and X_train is your training set
# Identify categorical columns in the DataFrame
categorical_columns =
d_new.select_dtypes(include=['object']).columns.tolist()

# Convert categorical columns to indices
categorical_features_indices = [X_train.columns.get_loc(col) for col in
categorical_columns]
```

```
# Define categorical names for LIME explainer
categorical_names = {i: d_new[col].unique().tolist() for i, col in
zip(categorical_features_indices, categorical_columns)}

# Create a LIME explainer
explainer = lime.lime_tabular.LimeTabularExplainer(
    training_data=X_train.values,
    mode='classification',
    training_labels=Y_train,
    feature_names=X_train.columns.tolist(),
    categorical_features=categorical_features_indices,
    categorical_names=categorical_names
)
```

- CELL-61

```
import xgboost as xgb
import lime.lime_tabular

# Assuming X_train, y_train are your training data
model = xgb.XGBClassifier()  # Change from XGBRegressor to
XGBClassifier
model.fit(X_train, Y_train)

# Create a LIME explainer
explainer = lime.lime_tabular.LimeTabularExplainer(
    training_data=X_train.values,
    mode='classification',
    training_labels=Y_train,
    feature_names=X_train.columns.tolist(),
    categorical_features=categorical_features_indices,
    categorical_names=categorical_names
)

# Choose an instance to explain
i = 0  # Index of the instance to explain
instance = X_test.iloc[i]  # Extract the instance

# Explain the instance's prediction
explanation = explainer.explain_instance(instance, model.predict_proba,
num_features=10)
```

```
# Display the explanation
explanation.show_in_notebook(show_all=False)
print(explanation.as_list())
```

- CELL-62

```
explanation.as_list()
```

CHAPTER 9

REFERENCE

1.“Credit Card Fraud Detection System” by K. Madkaikar, M. Nagvekar, P. Parab, R. Raikar, S. Patil.
2.“Credit Card Fraud Detection using Machine Learning Algorithms” by V. Dornadulaa, Geetha S.
3. “A Research Paper on Credit Card Fraud Detection” by S. Teja, B. Munendra, S. Gokulkrishnan.
4. “Credit Card Fraud Detection Predictive Modelling” by N. Sharma.
5. Research on Credit Card Fraud Detection Model Based on Distance Sum – by Wen-Fang YU and Na Wang” published by 2009.
6. International Joint Conference on Artificial Intelligence
Maniraj SP, Saini A, Ahmed S, Sarkar D. Credit card fraud detection using machine learning and data science. Int J Eng Res 2019.
7. Breiman L. Random forests. Mach Learning
Campus K. Credit card fraud detection using machine learning models and collating machine learning models. Int J Pure Appl Math.
8. A. Shen, R. Tong, and Y. Deng, "Application of classification models on credit card fraud detection," June 2007.
9. Dahl, J.: Card Fraud. In: Credit Union Magazine (2006). 10.Ghosh and D.L. Reilly, “Credit Card Fraud Detection with a Neural-Network,” Proc. 27th Hawaii International Conference on System Sciences: Information Systems: Decision Support and Knowledge-Based Systems, vol. 3, pp. 621-630, 1994.