

Credit Card Fraud Prevention: Exploratory Analysis and Predictive Models

Phase – 1

Teammates: Pavan Kalyan Reddy Bodugu, Venkata Sai Sunil Bhattar Paruchuri

UBIT Names: pbodugu, vparuchu

UBIT Numbers: 50532195, 50533606

Some Background:

In recent years, the global economy has witnessed a surge in digital transactions, driven by the increasing popularity of online shopping, mobile banking, and electronic payments. While this digital transformation has brought about unparalleled convenience and efficiency, it has also created new opportunities for fraudulent activities, particularly in the realm of financial transactions. According to a report by the Association of Certified Fraud Examiners (ACFE), organizations worldwide lose an estimated 5% of their annual revenues to fraud, amounting to trillions of dollars annually. Among various types of fraud, credit card fraud stands out as a significant concern, with sophisticated fraudsters employing ever-evolving techniques to exploit vulnerabilities in payment systems.

Problem Statement:

The overarching goal of this project is to develop a robust and accurate predictive model for detecting fraudulent transactions in real-time. Leveraging machine learning algorithms and advanced analytics, the model aims to analyze transaction data and identify patterns indicative of fraudulent behavior. By flagging suspicious transactions promptly, financial institutions and online merchants can mitigate financial losses, safeguard customer assets, and preserve the integrity of the financial ecosystem.

Background leading to our objective:

- **Digital Transformation:** The proliferation of digital transactions has created a fertile ground for fraudulent activities, necessitating the development of sophisticated fraud detection mechanisms to counteract evolving threats.

- **Rise in Cybercrime:** Cybercriminals continuously innovate and adapt their tactics to exploit vulnerabilities in payment systems, highlighting the importance of robust fraud detection solutions capable of keeping pace with emerging threats.
- **Regulatory Compliance:** Financial institutions and online merchants are subject to stringent regulatory requirements aimed at combating financial crime and protecting consumer interests, underscoring the need for effective fraud detection and prevention measures.

Potential of the Project:

- **Financial Loss Mitigation:** The predictive model has the potential to significantly reduce financial losses incurred by organizations due to fraudulent transactions, thereby safeguarding their revenues and profitability.
- **Enhanced Security:** By effectively detecting and preventing fraudulent activities, the project contributes to the overall security of digital transactions, fostering consumer confidence and trust in online payment systems.
- **Operational Efficiency:** Automated fraud detection streamlines the process of identifying suspicious transactions, enabling organizations to allocate their resources more efficiently and focus on strategic initiatives.
- **Trust of the Customers:** Enhancing customer trust is crucial, as effective fraud prevention measures instill confidence in cardholders regarding the security of their financial assets. When customers have faith in the safety of their transactions, they are more likely to utilize credit cards for their financial activities and develop long-term relationships with the bank.

Target User:

- **Financial Institutions:** Banks, credit card companies, and other financial institutions can utilize the predictive model to strengthen their fraud detection capabilities and proactively combat fraudulent activities.
- **Online Merchants:** E-commerce platforms and online retailers can integrate the model into their payment processing systems to protect their businesses and customers from fraudulent transactions, thereby enhancing trust and loyalty.

Data Sources:

The dataset utilized for our model was sourced from Kaggle, a well-regarded platform known for hosting a wide array of data science and machine learning resources, including datasets and models.

Link for the dataset: -

<https://www.kaggle.com/datasets/anurag629/credit-card-fraud-transaction-data/data>

Data Cleaning/ Processing:

We conducted data cleaning and preprocessing tasks to address inconsistencies found within the raw dataset. These procedures aimed to enhance the overall quality of the dataset and minimize potential biases.

Our initial step involved utilizing the `read_csv()` function from the pandas library to import the dataset's Excel file as a data frame, enabling us to execute subsequent operations.

1. Removing unwanted columns:

We removed the 'Day of Week' column from the data frame 'df' using the `drop()` function. This column was deemed unnecessary for our analysis and model development purposes, so we chose to eliminate it.

2. Standardization of Data:

We performed data standardization by converting the 'Date' column in the data frame 'df' to a datetime format using the `pd.to_datetime()` function. This allowed us to ensure uniformity and consistency in the date format across the dataset. Subsequently, we used the `dt.strftime()` method to standardize the date format to 'YYYY-MM-DD', facilitating easier interpretation and analysis of the data.

```
df['Date'] = pd.to_datetime(df['Date'])
df['Date'] = df['Date'].dt.strftime('%Y-%m-%d')
```

3. Merging Columns:

In this step, we merged the 'Date' and 'Time' columns in the data frame 'df' to create a single datetime column representing the complete timestamp of each transaction. First, we converted the 'Date' column to datetime format using `pd.to_datetime()`. Then, we converted the 'Time' column to timedelta format, specifying the unit as hours. Next, we added the 'Time' column to the 'Date' column to obtain the combined timestamp. Finally, we dropped the 'Time' column as it was no longer needed after merging.

```
df['Time'] = pd.to_timedelta(df['Time'], unit="h")
df['Date'] = df['Date'] + df['Time']
```

```
df = df.drop('Time', axis=1)
```

4. Renaming column Names:

The 'Shipping Address' column was renamed to 'Country of Shipping' to better reflect its content, and the 'Date' column was renamed to 'Date&Time' to accurately represent the merged timestamp.

```
df.rename(columns={'Shipping Address': 'Country of Shipping', 'Date': 'Date&Time'},  
inplace=True)
```

5. Handling improper text formatting:

In this process, we addressed improper text formatting in the 'Transaction ID', 'Amount', and 'Bank' columns of the dataframe 'df'. Firstly, we removed any '#' symbols and extra spaces from the 'Transaction ID' column to ensure uniformity. Secondly, we removed the currency symbol '£' from the 'Amount' column to facilitate numerical operations. Subsequently, we corrected a spelling inconsistency in the 'Bank' column by replacing instances of 'Barlcays' with 'Barclays'. This ensures consistency and accuracy in the dataset. Finally, we verified the changes made by displaying the unique values before and after updating the 'Bank' column.

```
df['Transaction ID'] = df['Transaction ID'].apply(lambda x: x.replace('#', ''))  
df['Transaction ID'] = df['Transaction ID'].apply(lambda x: x.replace(' ', ''))  
df['Amount'] = df['Amount'].str.replace('£', '')  
df['Bank'] = df['Bank'].apply(lambda x: "Barclays" if x == "Barlcays" else x)
```

6. Handling Data Types:

Firstly, we converted the 'Age' column values to integers to ensure numerical consistency. Secondly, we converted the 'Amount' column values to floating-point numbers to facilitate numerical calculations and analysis.

```
df['Age'] = [int(age) for age in df['Age']]  
df['Amount'] = df['Amount'].apply(float)
```

7. Handling null values:

Upon conducting a thorough examination for null values across all columns, we identified that the 'Amount', 'Merchant Group', 'Country of Shipping', and 'Gender' columns exhibit instances of missing data.

Merchant Group: Filled null values with the mode (most frequent value) of the 'Merchant Group' column to maintain data integrity.

Country of Shipping: Imputed missing values in the 'Country of Shipping' column with the mode of the column to ensure completeness of shipping information.

Gender: Dropped rows with null values in the 'Gender' column to maintain consistency and completeness of demographic data.

8. Imputation of Missing Values:

Amount: Addressed null values in the 'Amount' column by replacing them with 0 and visualized the distribution of non-null values using a histogram for insights.

9. Outliers:

Amount: Detected outliers using the Interquartile Range (IQR) method and visualized them through a boxplot. Outliers were identified as values falling outside 1.5 times the IQR from the quartiles.

```
Q1 = df['Amount'].quantile(0.25)
```

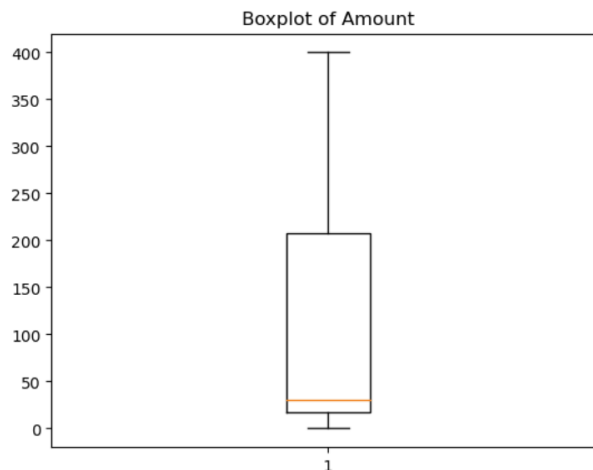
```
Q3 = df['Amount'].quantile(0.75)
```

```
IQR = Q3 - Q1
```

```
lower_bound = Q1 - 1.5 * IQR
```

```
upper_bound = Q3 + 1.5 * IQR
```

```
outliers = df[(df['Amount'] < lower_bound) | (df['Amount'] > upper_bound)]
```



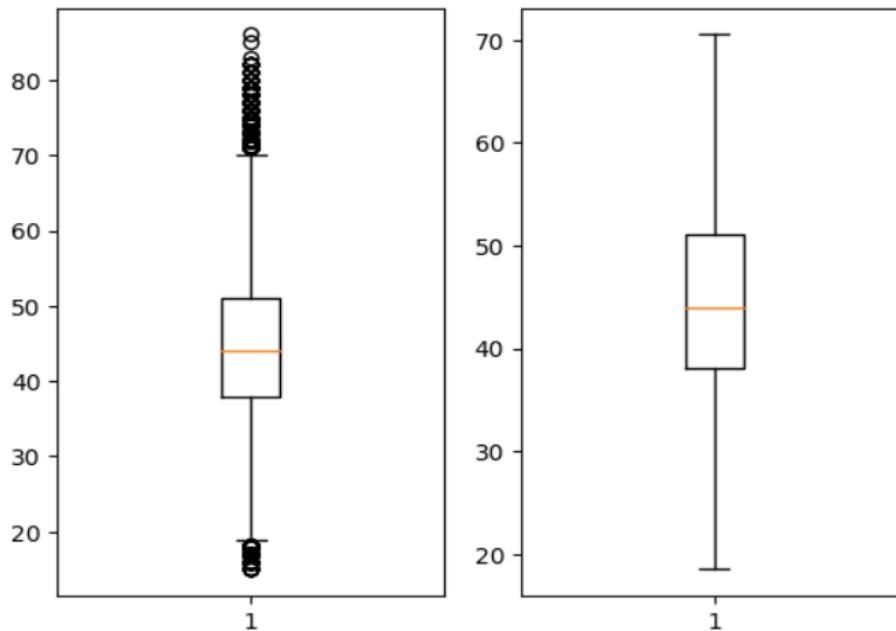
Upon examination, it appears that the 'Amount' column does not contain any outliers.

Age: Similarly, outliers in the 'Age' column were identified and treated by capping extreme values beyond 1.5 times the IQR, ensuring the integrity of the data distribution. Boxplots were employed for visualization before and after outlier treatment.

```
Q1_age = np.quantile(df['Age'], 0.25)
Q3_age = np.quantile(df['Age'], 0.75)
median_age = np.quantile(df['Age'], 0.5)
```

```
IQR_age = Q3_age - Q1_age
lower_bound_age = Q1_age - 1.5 * IQR_age
upper_bound_age = Q3_age + 1.5 * IQR_age
```

```
df['Age'] = np.where(df['Age'] > upper_bound_age, upper_bound_age, np.where(df['Age']
< lower_bound_age, lower_bound_age, df['Age']))
```



We observed outliers present in the 'Age' column, and to address this issue, we utilized the Interquartile Range (IQR) method to replace these outliers with more representative values.

10. Handling Duplicate values:

Duplicate values were addressed by removing them from the data frame 'df'. The original length of the data frame was compared to the length after duplicate removal to evaluate the effectiveness of the process.

11. Encoding:

Categorical columns were encoded using one-hot encoding technique to convert them into numerical representations, enhancing their compatibility for machine learning algorithms.

Additionally, the 'Gender' column was mapped to numerical values ('M': 0, 'F': 1) to facilitate further analysis and modeling.

We used label encoding for the 'Gender' column We used one hot encoding for ['Type of Card', 'Entry Mode', 'Type of Transaction', 'Merchant Group', 'Country of Residence', 'Country of Shipping', 'Country of Transaction', 'Bank'] columns.

Exploratory Data Analysis:

In this phase, our objective is to gain a deeper statistical understanding of the data by employing various visualization techniques. We will utilize different types of graphs and charts to extract valuable insights from the dataset, allowing us to uncover patterns, trends, and relationships within the data.

1. Summary Statistics of the Dataset:

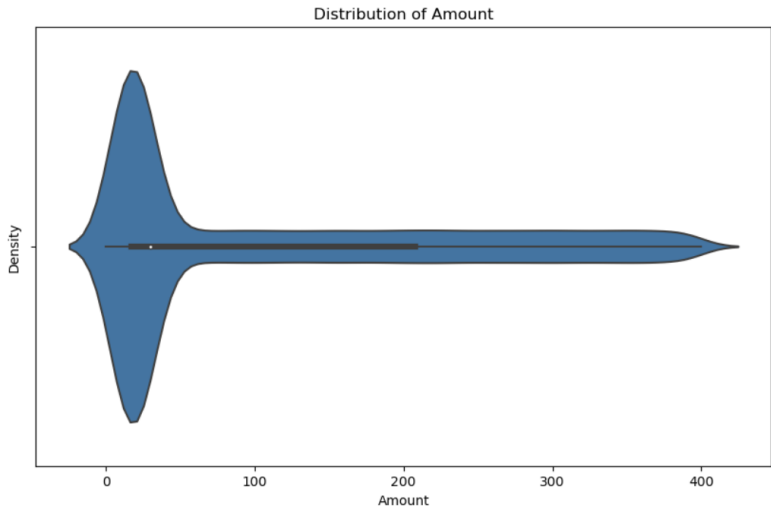
The describe() function provides a comprehensive summary of statistical measures such as mean, median, standard deviation, and quartiles for numerical columns in the dataset, offering insights into the central tendency, dispersion, and distribution of the data.

	Amount	Gender	Age	Fraud	Type of Card_MasterCard	Type of Card_Visa	Entry Mode_CVC	Entry Mode_PIN	Entry Mode_Tap	Type of Transaction_ATM	...	Country of Transaction_Russia	Country of Transaction_USA	Country of Transaction_United Kingdom	Bank_Barclays	Bank
count	99996.000000	99996.000000	99996.000000	99996.000000	99996.000000	99996.000000	99996.000000	99996.000000	99996.000000	99996.000000	...	99996.000000	99996.000000	99996.000000	99996.000000	99996.000000
mean	112.569983	0.491230	44.534021	0.071953	0.461878	0.538122	0.334803	0.499760	0.165437	0.332733	...	0.072563	0.072983	0.711988	0.399286	0.000000
std	123.433431	0.499926	9.896884	0.258411	0.498547	0.498547	0.471924	0.500002	0.371576	0.471194	...	0.259419	0.260110	0.452839	0.489754	0.000000
min	0.000000	0.000000	18.500000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000
25%	17.000000	0.000000	38.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000
50%	30.000000	0.000000	44.000000	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	1.000000	0.000000	0.000000
75%	208.000000	1.000000	51.000000	0.000000	1.000000	1.000000	1.000000	1.000000	0.000000	1.000000	...	0.000000	0.000000	1.000000	1.000000	0.000000
max	400.000000	1.000000	70.500000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...	1.000000	1.000000	1.000000	1.000000	1.000000

8 rows x 44 columns

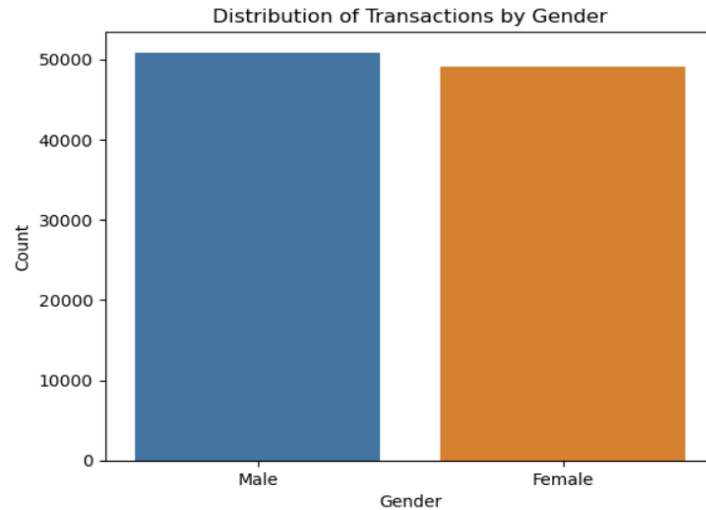
2. Violin Plot of Transaction Amount Distribution:

The violin plot illustrates the distribution of transaction amounts, providing insights into the density and variability of transaction values within the dataset. This graph indicates that the majority of transactions involve amounts below 50.



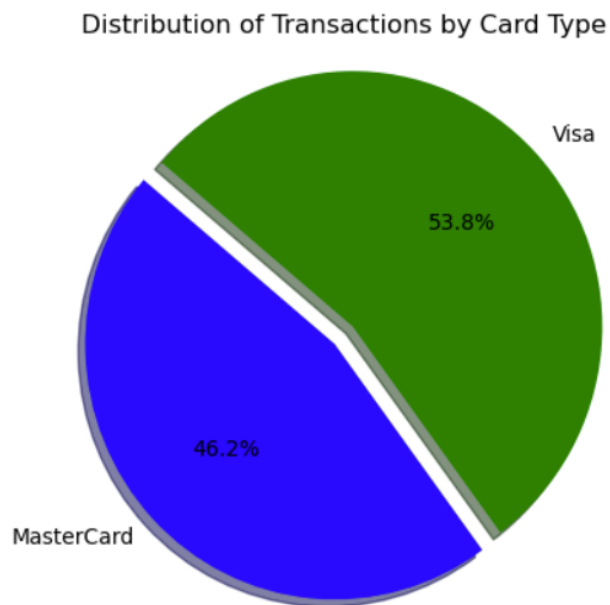
3. Distribution of Transactions by Gender:

The count plot visualizes the distribution of transactions categorized by gender, providing insights into the frequency of transactions made by male and female customers. The graph illustrates that the count of transactions attributed to males is higher than that of females, with a count of 50,000 on the y-axis.



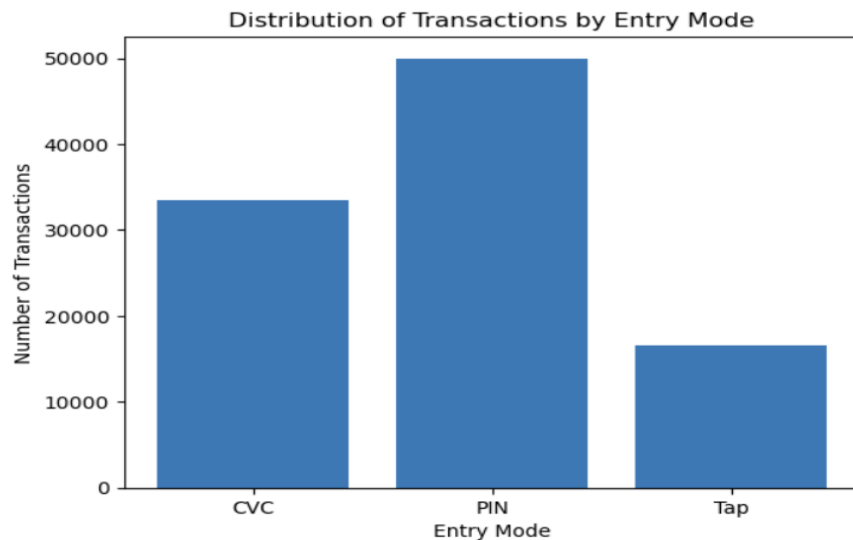
4. Distribution of Transactions by Card Type:

The pie chart illustrates the distribution of transactions between MasterCard and Visa, indicating that MasterCard accounts for 46.2% of transactions while Visa constitutes 53.8%. This suggests a slightly higher prevalence of Visa transactions compared to MasterCard within the dataset.



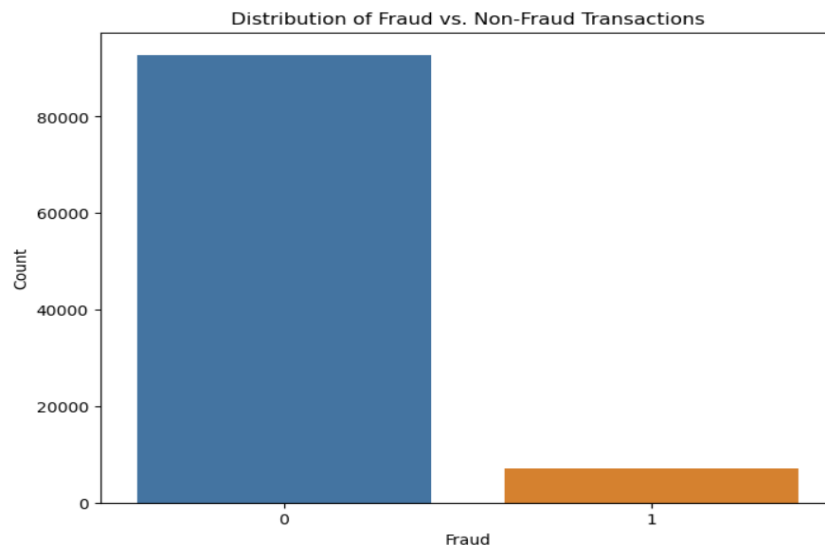
5. Distribution of Transactions by Entry Mode:

The bar graph compares the distribution of transactions across different entry modes: CVC, PIN, and Tap. It reveals that PIN transactions have the highest count, followed by CVC, while Tap transactions have the lowest count within the dataset, each representing a significant portion of the data. This suggests varied preferences in transaction entry methods among customers.



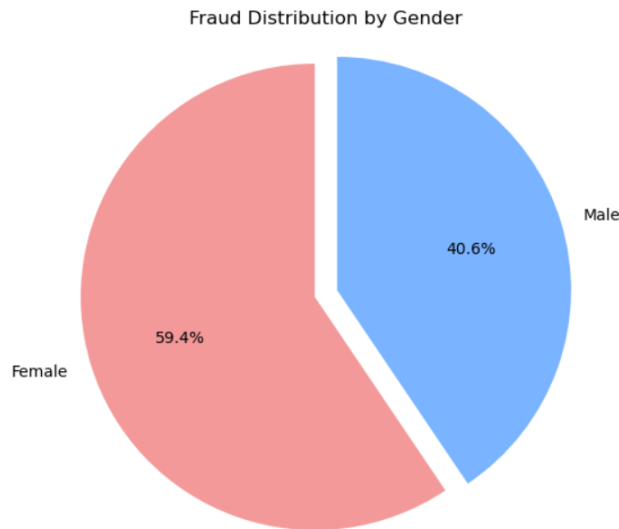
6. Distribution of Fraud vs. Non-Fraud Transactions:

The count plot visualizes the distribution of fraudulent and non-fraudulent transactions. It demonstrates the disparity in counts between the two categories, providing insights into the prevalence of fraud within the dataset. The graph indicates that the majority of transactions are classified as non-fraudulent, with only a small fraction identified as fraudulent transactions.



7. Fraud Distribution by Gender:

The pie chart illustrates the distribution of fraudulent transactions categorized by gender, indicating that female fraud accounts for 59.4% while male fraud constitutes 40.6% of the total fraudulent transactions. This suggests a slightly higher prevalence of fraud among female customers compared to male customers within the dataset.



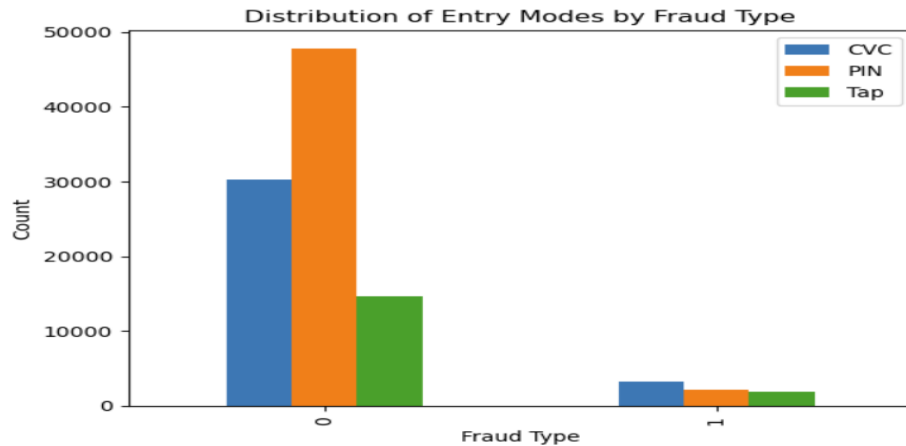
8. Distribution of Card Types by Fraud Type:

The bar chart compares the distribution of card types (MasterCard and Visa) across fraudulent and non-fraudulent transactions. In non-fraudulent transactions, Visa cards have a higher count compared to MasterCard. However, in fraudulent transactions, Visa cards also dominate with a higher count compared to MasterCard. This suggests that Visa cards are more prevalent in both fraudulent and non-fraudulent transactions within the dataset.



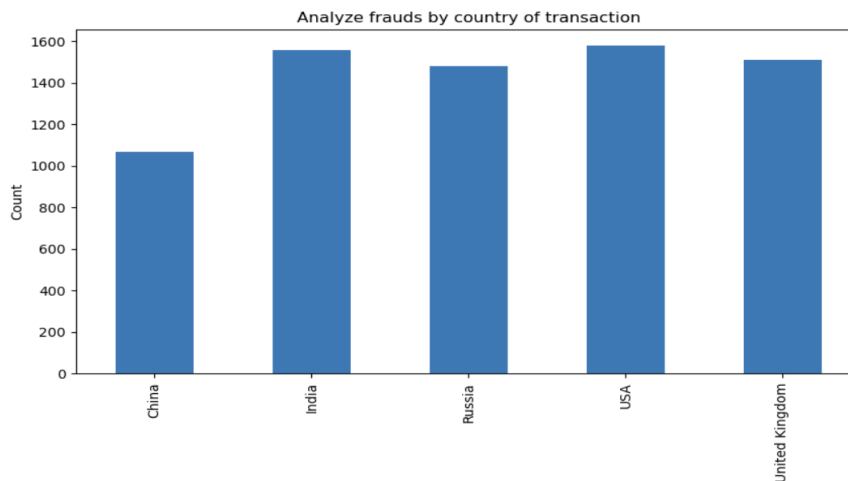
9. Distribution of Entry Modes by Fraud Type:

From the graph, we observe that non-fraudulent transactions predominantly occur via the PIN mode, whereas fraudulent transactions are primarily associated with the CVC mode, followed by PIN and Tap.



10. Fraud Distribution by Country of Transaction:

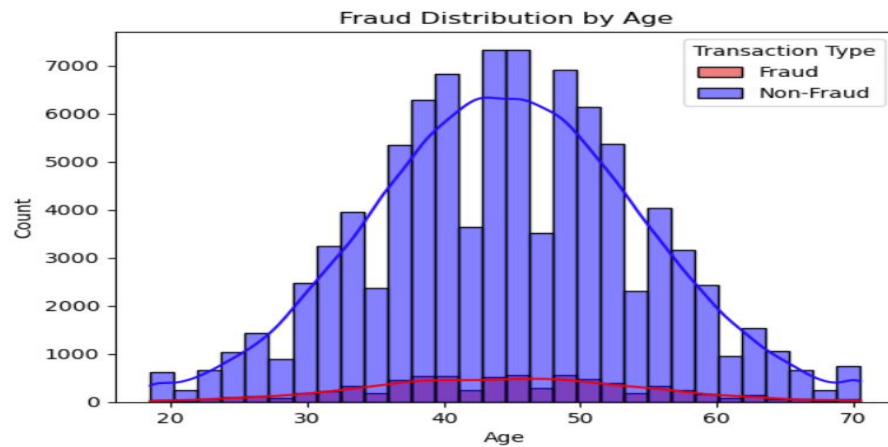
The bar chart examines fraud occurrences across different countries of transaction. China exhibits the lowest count of frauds, while India and the USA demonstrate similar levels of fraudulent activity. Russia and the United Kingdom also present comparable fraud counts, albeit lower than those observed in India and the USA.



11. Fraud Distribution by Age:

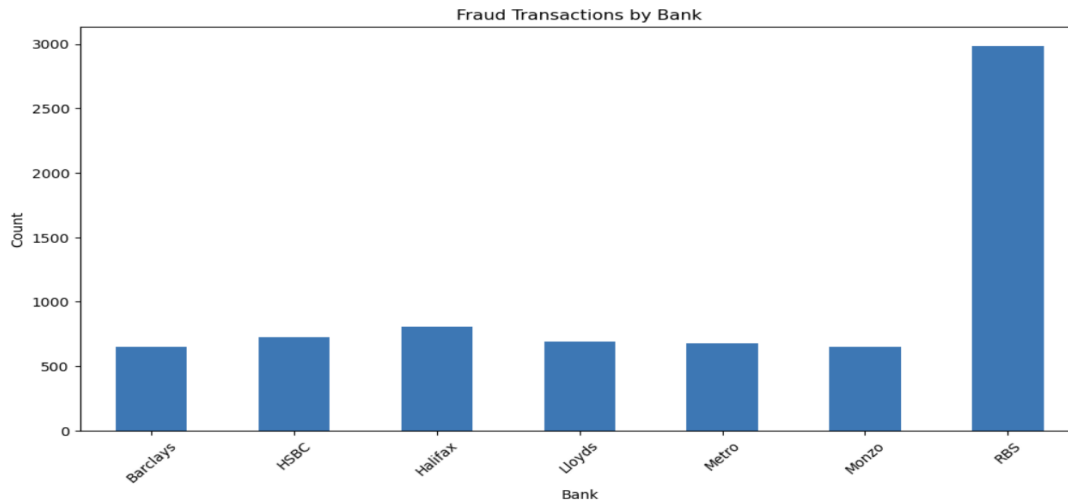
The histogram visualizes the distribution of fraud and non-fraud transactions categorized by age. Both fraudulent and non-fraudulent transactions exhibit a concentration between

the ages of 35 to 55, suggesting a similarity in age distribution between the two transaction types.



12. Fraud Transactions by Bank:

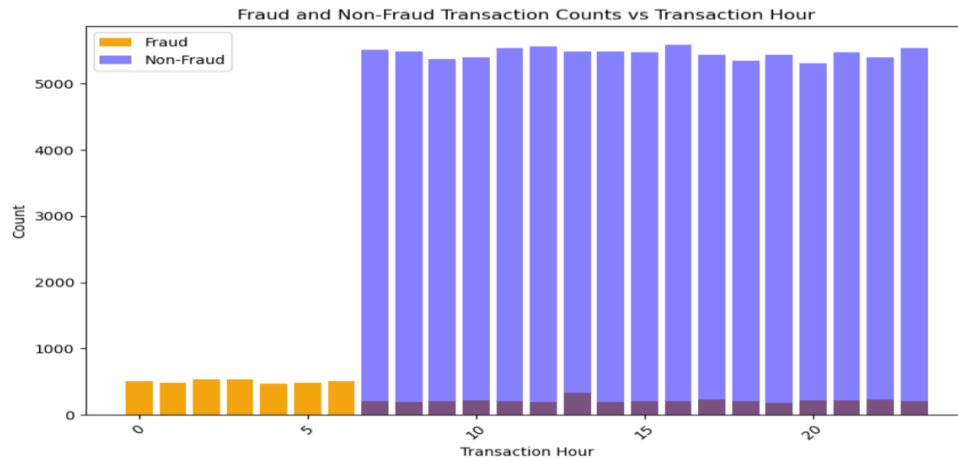
The bar chart displays the count of fraudulent transactions across different banks. Notably, RBS bank stands out with a significantly higher number of fraudulent transactions compared to other banks. This observation suggests a potential area of focus for fraud prevention efforts within RBS bank.



13. Fraud and Non-Fraud Transaction Counts vs Transaction Hour:

The bar chart illustrates the counts of fraudulent and non-fraudulent transactions across different hours of the day. It reveals that the majority of fraudulent transactions occur in the early morning hours, while there is a significant decrease in non-fraudulent transactions during this period. This observation underscores the importance of heightened vigilance and fraud detection measures, particularly during the early morning

hours, to mitigate fraudulent activities.



14. Correlation:

To assess the relationship between each attribute and the 'Fraud' feature, we calculated the correlation values. A higher absolute correlation value indicates a stronger effect of the attribute on the 'Fraud' feature.

Correlation Analysis with Fraud:

Country of Transaction_United Kingdom	-0.346523
Country of Shipping_United Kingdom	-0.337594
Transaction Hour	-0.286532
Entry Mode_PIN	-0.112212
Amount	-0.109875
Type of Transaction_ATM	-0.059218
Country of Residence_Russia	-0.055545
Country of Residence_USA	-0.055526
Country of Residence_China	-0.054692
Country of Residence_India	-0.053542
Gender	-0.047639
Type of Card_MasterCard	-0.034793
Merchant Group_Restaurant	-0.023792
Merchant Group_Subscription	-0.020386
Merchant Group_Products	-0.018824
Merchant Group_Gaming	-0.017705
Merchant Group_Food	-0.016953
Merchant Group_Services	-0.016467
Merchant Group_Entertainment	-0.015325
Bank_RBS	-0.009119
Bank_HSBC	-0.008190
Type of Transaction_POS	-0.005839
Bank_Halifax	-0.004981
Bank_Lloyds	-0.003102
Age	-0.002830
Bank_Monzo	0.000136

Bank_Barclays	0.008782
Bank_Metro	0.010908
Merchant Group_Fashion	0.026512
Type of Card_Visa	0.034793
Merchant Group_Electronics	0.037227
Merchant Group_Children	0.064013
Entry Mode_CVC	0.064955
Type of Transaction_Online	0.064955
Entry Mode_Tap	0.068498
Country of Residence_United Kingdom	0.118425
Country of Shipping_Russia	0.133689
Country of Shipping_China	0.136943
Country of Shipping_India	0.142055
Country of Shipping_USA	0.142092
Country of Transaction_Russia	0.143050
Country of Transaction_China	0.149484
Country of Transaction_USA	0.153825
Country of Transaction_India	0.160776
Fraud	1.000000

15. Normalization:

Here we employed data normalization techniques to preprocess our dataset, ensuring optimal performance of subsequent analytical tasks. Specifically, we focused on normalizing the 'Age' and 'Amount' columns using Min-Max scaling. This technique rescales the features to a standardized range, typically between 0 and 1, thereby eliminating potential biases arising from disparate scales across features. To achieve this, we leveraged the `MinMaxScaler` module from the `scikit-learn` library, a widely-used tool in machine learning pipelines. By applying Min-Max scaling, we transformed the 'Age' and 'Amount' features to conform to a uniform scale, facilitating fair and accurate comparisons between data points.

```
columns_to_normalize = ['Age', 'Amount']
scaler = MinMaxScaler()
```

Phase – 2

1. Logistic Regression:

Reason for Algorithm Choice on Credit Card Fraud Transactions:

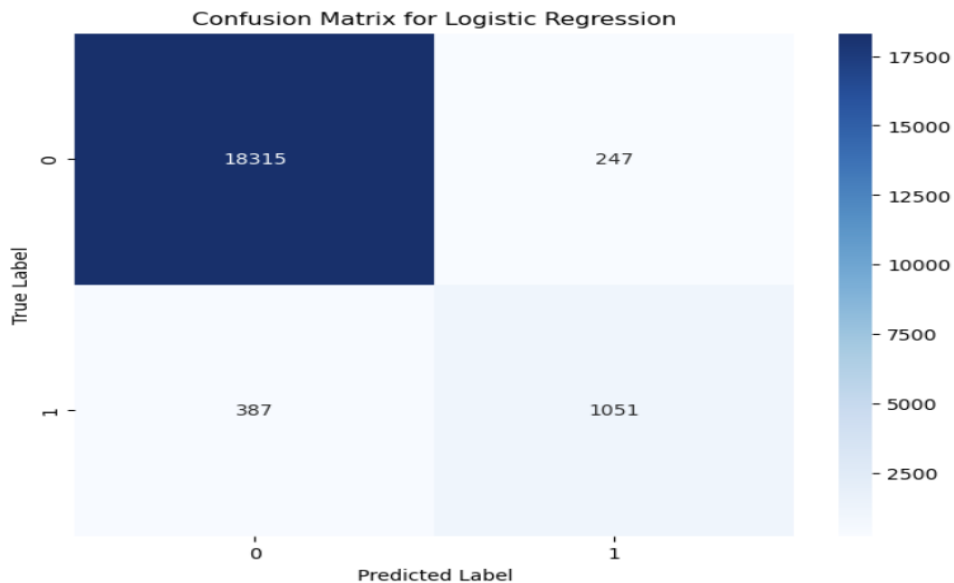
- Logistic regression was chosen for its simplicity, interpretability, and effectiveness in binary classification tasks. Given the nature of credit card fraud detection, where the goal is to classify transactions as fraudulent or not, logistic regression provides a straightforward approach. Additionally, logistic regression allows us to understand the impact of individual features on the likelihood of fraud, which is crucial for interpretability and risk assessment in financial transactions.

Classification Report:

Classification Report for Logistic Regression:				
	precision	recall	f1-score	support
Not Fraud	0.98	0.99	0.98	18562
Fraud	0.81	0.73	0.77	1438
accuracy			0.97	20000
macro avg	0.89	0.86	0.88	20000
weighted avg	0.97	0.97	0.97	20000

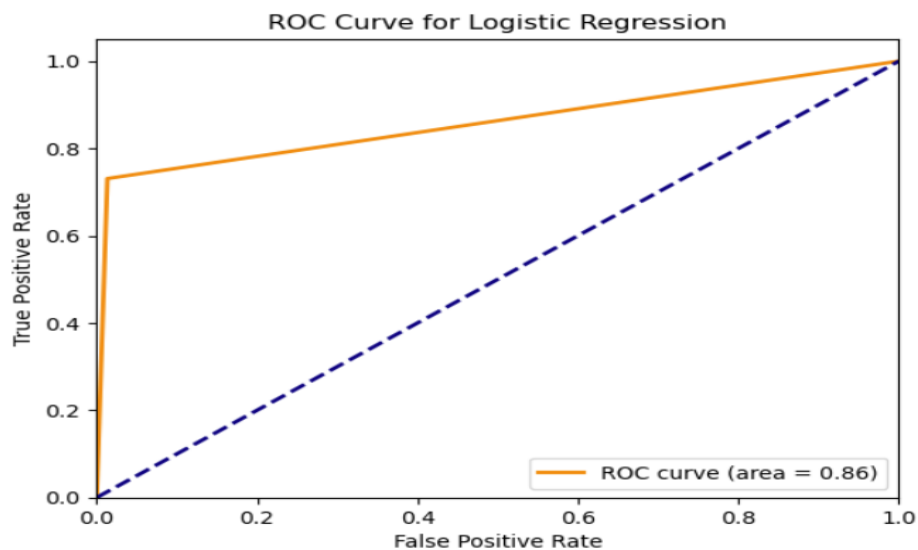
- Precision: The precision for the "Fraud" class is 0.81, indicating that among all transactions predicted as fraudulent, 81% were actually fraudulent.
- Recall: The recall for the "Fraud" class is 0.73, meaning that the model correctly identified 73% of all actual fraudulent transactions.
- F1-score: The F1-score for the "Fraud" class is 0.77, which is the harmonic mean of precision and recall. It provides a balanced measure of the model's performance on the "Fraud" class.
- Accuracy: The overall accuracy of the model is 0.97, indicating that it correctly classified 97% of all transactions.
- Macro Avg/Weighted Avg: These metrics provide averages across all classes, weighted or not, depending on the chosen averaging strategy.

Confusion Matrix:



- The confusion matrix shows that out of 18,562 non-fraudulent transactions, the model correctly classified 18,315 as non-fraudulent (True Negatives), but misclassified 247 as fraudulent (False Positives).
- Out of 1,438 fraudulent transactions, the model correctly classified 1,051 as fraudulent (True Positives), but misclassified 387 as non-fraudulent (False Negatives).

ROC Curve:



- The ROC curve plots the True Positive Rate (sensitivity) against the False Positive Rate (1 - specificity) for different threshold values.
- The area under the ROC curve (AUC) is a measure of the model's ability to distinguish between classes. An AUC of 0.86 indicates good discriminative power of the model, with higher values suggesting better performance.

Insights:

- The logistic regression model performs well in distinguishing between fraudulent and non-fraudulent transactions, achieving high accuracy and robustness.
- The model's high precision indicates that the majority of transactions predicted as fraudulent are indeed fraudulent, minimizing the risk of false alarms.
- The model's recall, although slightly lower than precision, still captures a substantial portion of actual fraudulent transactions, indicating its effectiveness in identifying fraudulent activity.
- The AUC value of 0.86 suggests that the logistic regression model has strong discriminative power and is capable of distinguishing between the two classes effectively.
- Overall, the logistic regression model is well-suited for credit card fraud detection, providing a balance between simplicity, interpretability, and performance.

Effectiveness of Logistic Regression Algorithm for Credit Card Fraud Detection:

The logistic regression algorithm proves to be highly effective for credit card fraud detection, showcasing strong performance metrics such as precision, recall, and accuracy. With a precision of 0.81 and a recall of 0.73, the model adeptly identifies fraudulent transactions while minimizing false positives. Its interpretability and robustness further enhance its utility in risk assessment and decision-making processes within financial institutions. Moreover, the model's discriminative power, as evidenced by an AUC of 0.86, underscores its ability to distinguish between fraudulent and non-fraudulent transactions effectively. Overall, logistic regression emerges as a reliable and efficient solution for addressing the challenge of credit card fraud, offering a valuable tool to safeguard against financial losses and maintain customer trust.

2. Naive Bayes:

Reason for Algorithm Choice on Credit Card Fraud Transactions:

Naive Bayes was selected for credit card fraud detection due to its simplicity, efficiency, and ability to handle large datasets with high dimensionality. Key reasons for choosing Naive Bayes include:

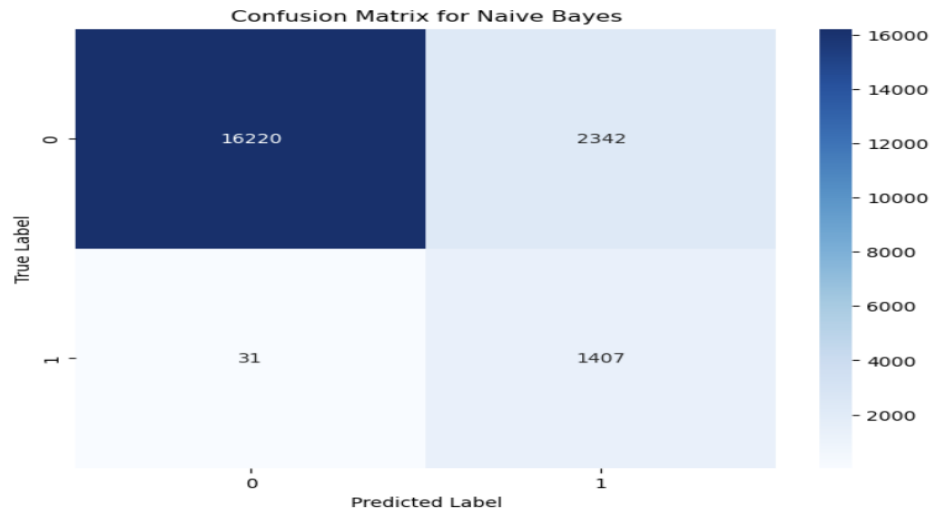
- **Efficiency:** Naive Bayes is computationally efficient and requires minimal tuning, making it suitable for processing large volumes of credit card transaction data in real-time.
- **Simple Assumption:** The Naive Bayes algorithm assumes independence between features, which, although oversimplified, can be effective for categorical data and has shown to work reasonably well in practice for fraud detection tasks.
- **Interpretability:** Naive Bayes provides clear probabilities for each class, making it easy to interpret the model's predictions and understand the likelihood of a transaction being fraudulent or non-fraudulent.

Classification Report:

Classification Report for Naive Bayes:				
	precision	recall	f1-score	support
Not Fraud	1.00	0.87	0.93	18562
Fraud	0.38	0.98	0.54	1438
accuracy			0.88	20000
macro avg	0.69	0.93	0.74	20000
weighted avg	0.95	0.88	0.90	20000

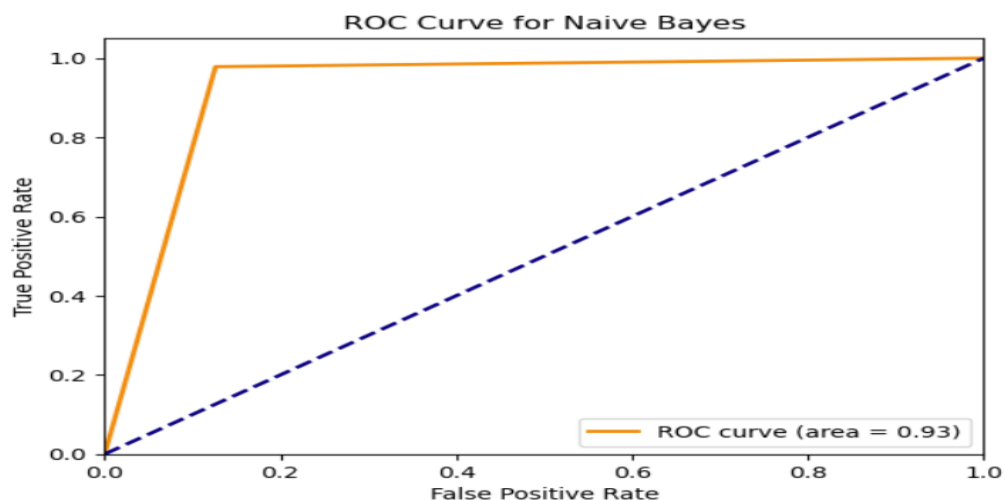
- Precision: 1.00 for "Not Fraud" and 0.38 for "Fraud". For "Not Fraud", it indicates that among all transactions predicted as not fraudulent, 100% were actually not fraudulent. For "Fraud", only 38% of predicted fraudulent transactions were actually fraudulent.
- Recall: 0.87 for "Not Fraud" and 0.98 for "Fraud". It means that the model correctly identified 87% of all actual non-fraudulent transactions and 98% of all actual fraudulent transactions.
- F1-Score: 0.93 for "Not Fraud" and 0.54 for "Fraud". F1-score is the harmonic mean of precision and recall, providing a balance between them.
- Accuracy: Overall accuracy is 0.88, indicating that the model correctly classified 88% of all transactions.

Confusion Matrix:



- True negatives (TN) represent the number of non-fraudulent transactions correctly classified by the model as non-fraudulent, with a count of 16,220 in this case.
- False positives (FP) indicate the number of non-fraudulent transactions incorrectly classified as fraudulent, with a count of 2,342.
- False negatives (FN) signify the number of fraudulent transactions incorrectly classified as non-fraudulent, with a count of 31.
- True positives (TP) denote the number of fraudulent transactions correctly classified by the model as fraudulent, with a count of 1,407.
- Analyzing these values helps evaluate the model's ability to accurately classify fraudulent and non-fraudulent transactions, highlighting areas for improvement, such as reducing false positives and false negatives.

ROC Curve:



- The ROC curve portrays the balance between sensitivity (true positive rate) and specificity (true negative rate) across various threshold values.
- A higher AUC (Area Under the Curve) value signifies better discriminative power of the model, where values closer to 1 indicate superior performance in distinguishing between fraudulent and non-fraudulent transactions.
- In this case, the AUC of 0.93 indicates the model's good discriminative power, suggesting its effectiveness in differentiating between fraudulent and non-fraudulent transactions with high accuracy and reliability.

Insights:

- Naive Bayes shows high precision for "Not Fraud" but relatively low precision for "Fraud", indicating a high rate of false positives for the "Fraud" class.
- It demonstrates high recall for both classes, especially for "Fraud", suggesting it effectively captures most of the actual fraudulent transactions.
- The model achieves a high AUC score of 0.93, indicating excellent discriminative power in distinguishing between fraudulent and non-fraudulent transactions.
- While Naive Bayes performs well in certain aspects, its relatively low precision for "Fraud" suggests a need for further optimization to reduce false positives and enhance overall performance.

Effectiveness of Naive Bayes Algorithm for Credit Card Fraud Detection:

Naive Bayes demonstrates effectiveness in certain aspects of credit card fraud detection:

- **High Recall:** Naive Bayes achieves a high recall of 0.98 for the "Fraud" class, indicating its ability to effectively capture most of the actual fraudulent transactions. This is crucial for minimizing the number of undetected fraudulent activities.
- **Good Discriminative Power:** The ROC curve's AUC score of 0.93 indicates that the Naive Bayes model has strong discriminative power and can effectively distinguish between fraudulent and non-fraudulent transactions.
- **Efficiency and Scalability:** Naive Bayes is computationally efficient and can handle large volumes of transaction data with high dimensionality, making it suitable for real-time credit card fraud detection systems.

However, Naive Bayes also shows limitations, particularly in precision for the "Fraud" class, where it achieves a relatively low precision of 0.38. This suggests a higher rate of false positives, which could lead to legitimate transactions being incorrectly flagged as fraudulent.

3. Support Vector Machine:

Reason for Algorithm Choice on Credit Card Fraud Transactions:

Support Vector Machine (SVM) with an RBF kernel was chosen due to its ability to handle non-linear decision boundaries and high-dimensional data effectively. Key reasons for selecting SVM include:

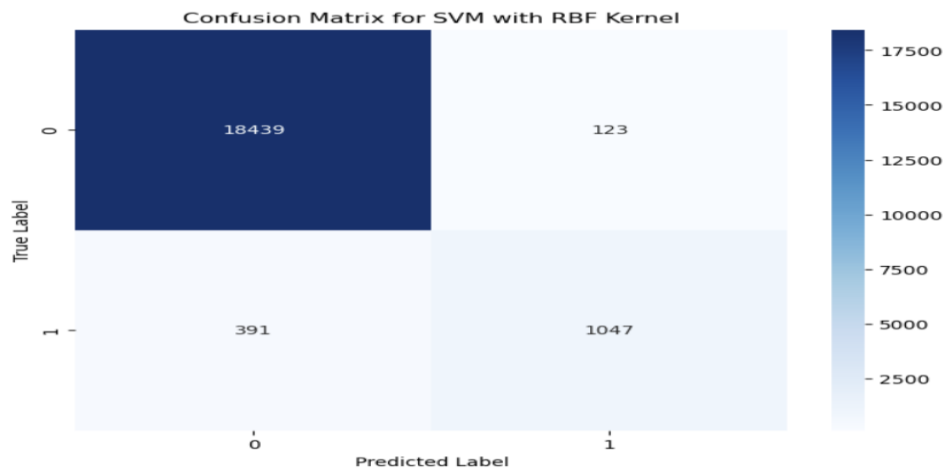
- **Non-linear Relationships:** SVM with an RBF kernel can capture complex, non-linear relationships in the data, which is essential for detecting intricate patterns indicative of credit card fraud.
- **Robustness to Outliers:** SVM is less sensitive to outliers compared to some other algorithms, making it suitable for datasets where fraudulent transactions may be outliers.
- **High-Dimensional Data:** SVM performs well in high-dimensional spaces, making it suitable for credit card transaction data, which typically involves a large number of features.

Classification Report:

Classification Report for SVM with RBF Kernel:				
	precision	recall	f1-score	support
Not Fraud	0.98	0.99	0.99	18562
Fraud	0.89	0.73	0.80	1438
accuracy			0.97	20000
macro avg	0.94	0.86	0.89	20000
weighted avg	0.97	0.97	0.97	20000

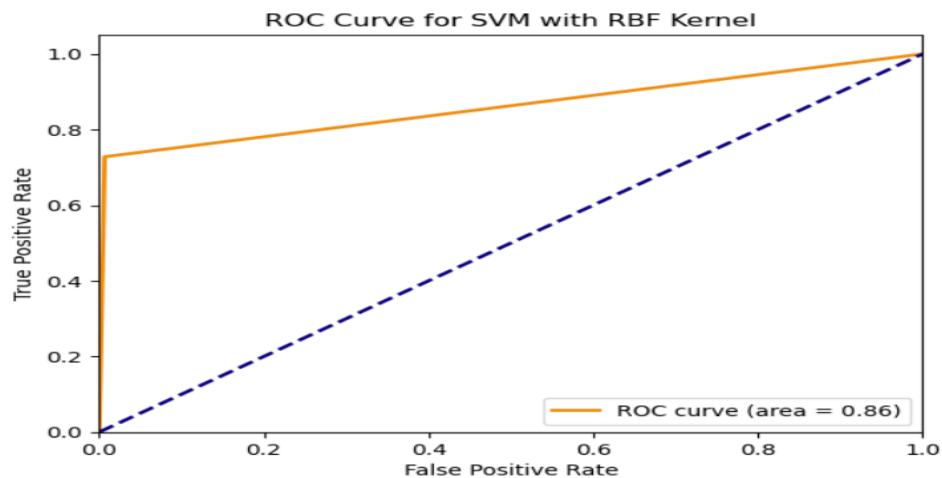
- The SVM with RBF Kernel model demonstrates strong performance metrics. It achieves a high precision of 0.98 for the "Not Fraud" class and 0.89 for the "Fraud" class, indicating that the majority of transactions classified as fraudulent are indeed fraudulent.
- The recall for the "Fraud" class is 0.73, suggesting that the model successfully identifies 73% of all actual fraudulent transactions.
- The F1-score, a harmonic mean of precision and recall, is 0.80 for the "Fraud" class, indicating a balanced performance between precision and recall.
- The overall accuracy of the model is 0.97, reflecting its ability to correctly classify 97% of all transactions.

Confusion Matrix:



- The confusion matrix provides a detailed breakdown of the model's performance. It shows that out of 18,562 non-fraudulent transactions, the model correctly classified 18,439 as non-fraudulent (True Negatives) and 123 as fraudulent (False Positives).
- For the 1,438 fraudulent transactions, the model correctly classified 1,047 as fraudulent (True Positives) and 391 as non-fraudulent (False Negatives).

ROC Curve:



- The ROC curve illustrates the trade-off between the true positive rate (sensitivity) and the false positive rate (1-specificity) across different threshold values.
- The ROC curve area (AUC) is 0.86, indicating good discriminative power of the model in distinguishing between fraudulent and non-fraudulent transactions.

Insights:

- SVM with an RBF kernel demonstrates robust performance in classifying both fraudulent and non-fraudulent transactions with high precision and recall.
- The model's high accuracy and AUC score suggest its effectiveness in distinguishing between the two classes, making it a valuable tool for credit card fraud detection.

- However, there is room for improvement in recall for the "Fraud" class, which could be further optimized to capture more fraudulent transactions while minimizing false negatives.

Effectiveness of SVM Algorithm for Credit Card Fraud Detection:

The Support Vector Machine (SVM) algorithm with an RBF kernel exhibits strong effectiveness in credit card fraud detection. With an overall accuracy of 97%, the model demonstrates robust performance in classifying both fraudulent and non-fraudulent transactions. Its high precision of 0.98 for non-fraudulent transactions and 0.89 for fraudulent ones underscores its ability to accurately identify instances of fraud. The SVM's AUC score of 0.86 reflects its capability to distinguish between fraudulent and non-fraudulent transactions with a high degree of precision.

4. K Means:

Reason for Algorithm Choice on Credit Card Fraud Transactions:

We selected the K Means Algorithm to evaluate the performance of our clustering algorithm with our dataset. K-means is a prominent unsupervised clustering technique designed to identify underlying patterns in data. By applying K-means to our credit card default data, we aimed to uncover hidden structures or groupings within the dataset, potentially representing distinct segments of credit card users. Once individual data points were assigned cluster labels, these labels could be utilized as additional features in a binary classification model. This integration of cluster information serves to enhance the classifier's performance by capturing complex relationships between the features and the target variable.

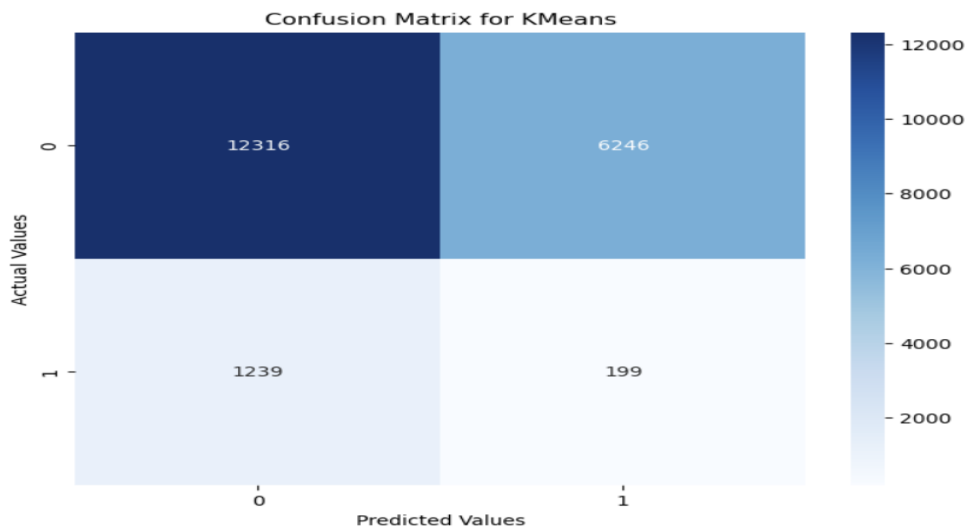
Classification Report:

Classification Report for Kmeans:				
	precision	recall	f1-score	support
Not Fraud	0.91	0.66	0.77	18562
Fraud	0.03	0.14	0.05	1438
accuracy			0.63	20000
macro avg	0.47	0.40	0.41	20000
weighted avg	0.85	0.63	0.72	20000

The classification report provides a summary of the precision, recall, F1-score, and support for each class (fraud and not fraud) based on K-means clustering:

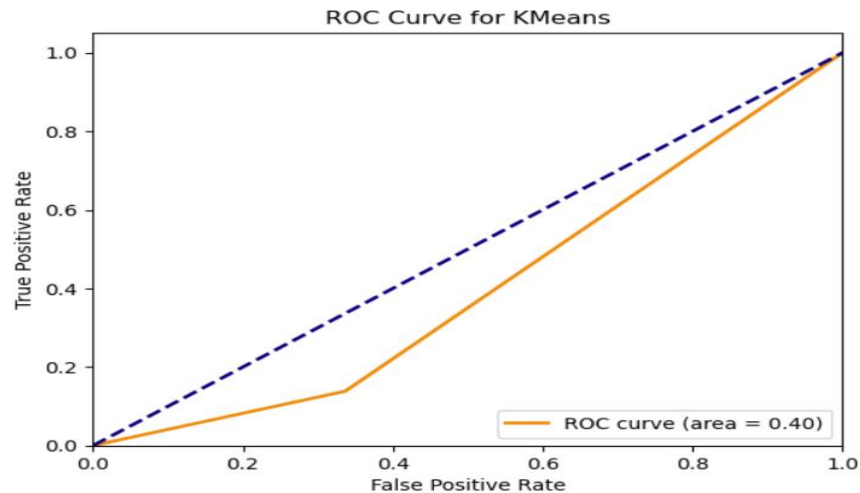
- **Precision:** Precision measures the proportion of true positive predictions among all instances predicted as positive. For fraudulent transactions, K-means achieved a precision of 0.03, indicating that only 3% of the instances classified as fraudulent were actually fraudulent.
- **Recall:** Recall, also known as sensitivity, measures the proportion of actual positive instances that were correctly identified by the model. In this case, the recall for fraudulent transactions is 0.14, indicating that K-means identified only 14% of all actual fraudulent transactions.
- **F1-score:** The F1-score is the harmonic mean of precision and recall, providing a balanced measure of the model's performance. The low F1-score of 0.05 for fraudulent transactions highlights the algorithm's poor performance in accurately classifying fraudulent instances.
- **Accuracy:** The overall accuracy of the K-means model is 0.63, indicating that it correctly classified 63% of all transactions. However, accuracy alone can be misleading, especially in imbalanced datasets where the majority class dominates.

Confusion Matrix:



- **True Negatives (TN):** These are the correctly classified non-fraudulent transactions. K-means correctly identified 12,316 instances as non-fraudulent.
- **False Positives (FP):** These are non-fraudulent transactions that were incorrectly classified as fraudulent. K-means misclassified 6,246 non-fraudulent transactions as fraudulent.
- **False Negatives (FN):** These are fraudulent transactions that were incorrectly classified as non-fraudulent. K-means misclassified 1,239 fraudulent transactions as non-fraudulent.
- **True Positives (TP):** These are the correctly classified fraudulent transactions. K-means correctly identified only 199 instances as fraudulent.

ROC Curve:



- The ROC curve visualizes the trade-off between the true positive rate and false positive rate at various threshold values. The area under the ROC curve (AUC) quantifies the algorithm's ability to distinguish between the two classes. In this case, the ROC curve's AUC value of 0.40 indicates relatively poor discriminative power compared to other algorithms.

Insights:

- The confusion matrix and classification report reveal significant challenges faced by K-means clustering in effectively identifying fraudulent transactions. The model's low precision, recall, and F1-score for fraudulent transactions suggest that it struggles to accurately classify instances belonging to the minority class.
- K-means clustering, being an unsupervised learning algorithm, lacks the ability to learn from labeled data and may not be well-suited for binary classification tasks like credit card fraud detection. Its reliance on distance-based clustering may lead to suboptimal performance, especially in datasets with complex distributions or imbalanced class distributions.
- While K-means clustering can provide insights into potential patterns or groupings within the data, it requires careful consideration of its limitations and may need to be supplemented with other techniques for effective fraud detection in real-world scenarios.

Effectiveness of KMeans Algorithm for Credit Card Fraud Detection:

The KMeans algorithm demonstrated limited effectiveness for credit card fraud detection. With a precision of 0.03 and recall of 0.14 for the "Fraud" class, it struggled to accurately identify fraudulent transactions. The confusion matrix revealed a high number of false positives (6246) and false negatives (1239), indicating significant misclassifications. The low ROC curve area of 0.40 further highlights the algorithm's poor discriminative power. Overall, KMeans showed suboptimal performance in detecting credit card fraud, suggesting its limitations in this context.

5. Random Forest Classifier:

Reason for Algorithm Choice on Credit Card Fraud Transactions:

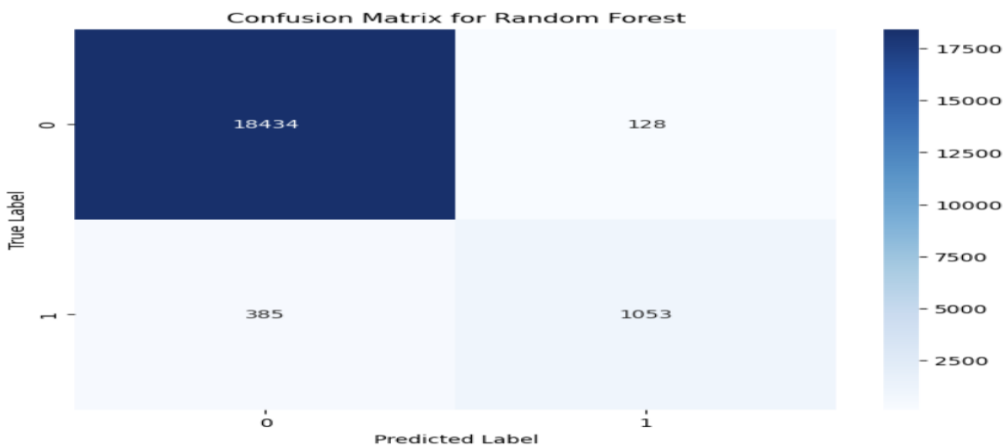
Random Forest was chosen due to its capability to handle large datasets with high dimensionality, making it suitable for credit card fraud detection, which often involves numerous features. Its ensemble learning approach, combining multiple decision trees, enhances model robustness and reduces overfitting, crucial for accurate fraud detection.

Classification Report:

Classification Report for Random Forest:					
	precision	recall	f1-score	support	
Not Fraud	0.98	0.99	0.99	18562	
Fraud	0.89	0.73	0.80	1438	
accuracy			0.97	20000	
macro avg	0.94	0.86	0.90	20000	
weighted avg	0.97	0.97	0.97	20000	

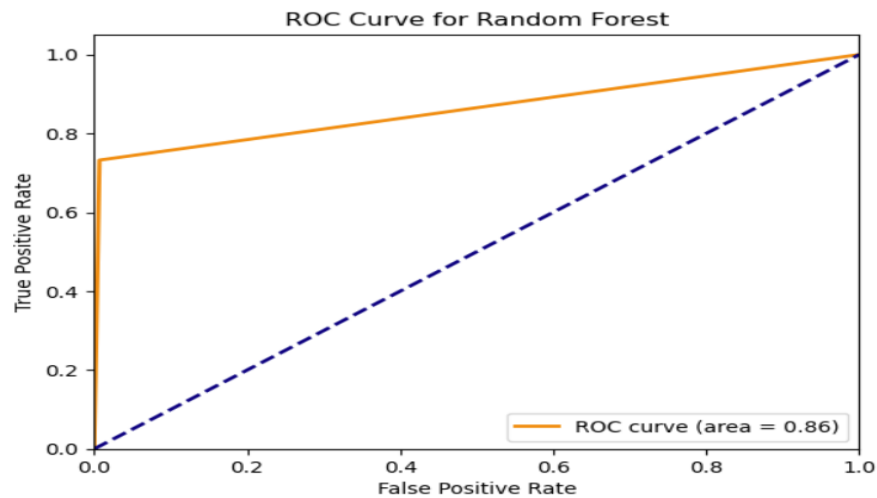
- Precision: The precision for the "Not Fraud" class is 0.98, indicating that among all transactions predicted as not fraudulent, 98% were actually not fraudulent. For the "Fraud" class, the precision is 0.89, meaning that among all transactions predicted as fraudulent, 89% were actually fraudulent.
- Recall: The recall for the "Not Fraud" class is 0.99, indicating that the model correctly identified 99% of all actual non-fraudulent transactions. For the "Fraud" class, the recall is 0.73, meaning that the model correctly identified 73% of all actual fraudulent transactions.
- F1-score: The F1-score provides a balance between precision and recall. For the "Not Fraud" class, the F1-score is 0.99, and for the "Fraud" class, it is 0.80.
- Accuracy: The overall accuracy of the model is 0.97, indicating that it correctly classified 97% of all transactions.

Confusion Matrix:



- The confusion matrix reveals that out of 18,562 non-fraudulent transactions, the model correctly classified 18,434 as non-fraudulent (True Negatives), with 128 misclassified as fraudulent (False Positives). Additionally, out of 1,438 fraudulent transactions, the model correctly identified 1,053 as fraudulent (True Positives), with 385 misclassified as non-fraudulent (False Negatives).

ROC Curve:



- The ROC curve shows an area under the curve (AUC) of 0.86, indicating good discriminative power of the Random Forest model in distinguishing between fraudulent and non-fraudulent transactions. A higher AUC value suggests better model performance, further validating the effectiveness of Random Forest for credit card fraud detection.

Insights:

- The Random Forest algorithm showcases robust performance in distinguishing between fraudulent and non-fraudulent transactions, as evidenced by its high precision, recall, and accuracy metrics.
- Its ensemble learning approach, leveraging multiple decision trees, enhances model stability and generalization, crucial for handling complex and large-scale datasets commonly found in credit card transactions.
- The model's ability to accurately identify fraudulent transactions while minimizing false positives is essential for financial institutions to mitigate risks and protect against potential losses.
- By effectively balancing sensitivity and specificity, the Random Forest algorithm proves to be a valuable tool for detecting credit card fraud, offering a reliable means of safeguarding financial transactions, and maintaining customer trust.

- Overall, the Random Forest algorithm demonstrates significant effectiveness in addressing the challenges associated with credit card fraud detection, making it a preferred choice for financial institutions seeking robust and accurate fraud detection solutions.

Effectiveness of Random Forest Classifier Algorithm for Credit Card Fraud Detection:

The Random Forest algorithm proves highly effective for credit card transaction detection, showcasing robust performance metrics across precision, recall, and F1-score. With a precision of 0.98 for non-fraudulent transactions and 0.89 for fraudulent transactions, the model accurately identifies the majority of both classes. Its high recall of 0.99 for non-fraudulent transactions ensures that it captures almost all actual non-fraudulent instances, while maintaining a respectable recall of 0.73 for fraudulent transactions. The model's overall accuracy of 97% underscores its capability to correctly classify the vast majority of transactions, instilling confidence in its reliability for fraud detection tasks. Overall, the Random Forest algorithm emerges as a formidable tool for safeguarding against fraudulent activities in credit card transactions, offering a potent solution for enhancing security and preserving trust in financial systems.

6. Gradient Boosting Machine:

Reason for Algorithm Choice on Credit Card Fraud Transactions:

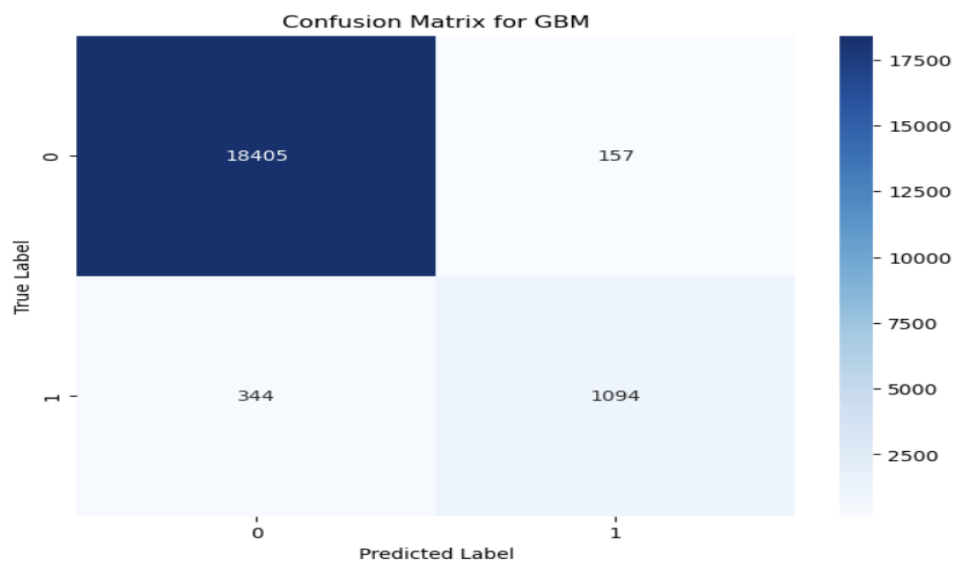
Gradient Boosting Machines (GBM) were selected for credit card fraud transactions due to their capability to handle complex, non-linear relationships within the data and their proven effectiveness in classification tasks. GBM works by sequentially adding weak learners to the model, each one focusing on the errors made by the previous learners, thereby improving the overall model performance. In the context of credit card fraud detection, where the patterns of fraudulent activities can be intricate and subtle, GBM's ability to iteratively refine the model's predictions makes it a suitable choice. Additionally, GBM is less prone to overfitting compared to other algorithms, ensuring robust performance on unseen data. Its flexibility in handling heterogeneous data types and its inherent feature importance analysis also contribute to its suitability for identifying fraudulent transactions accurately. Therefore, GBM was deemed appropriate for the task of credit card fraud detection, offering a balance between complexity, interpretability, and performance.

Classification Report:

Classification Report for Gradient Boosting Machines (GBM):					
	precision	recall	f1-score	support	
Not Fraud	0.98	0.99	0.99	18562	
Fraud	0.87	0.76	0.81	1438	
accuracy			0.97	20000	
macro avg	0.93	0.88	0.90	20000	
weighted avg	0.97	0.97	0.97	20000	

- The precision for non-fraudulent transactions is 0.98, indicating a high proportion of correctly classified non-fraudulent instances. For fraudulent transactions, the precision is 0.87, implying that 87% of the transactions predicted as fraudulent were indeed fraudulent.
- The recall for non-fraudulent transactions is 0.99, indicating that the model captures the vast majority of actual non-fraudulent instances. However, for fraudulent transactions, the recall is 0.76, suggesting that the model identifies only 76% of actual fraudulent instances.
- The F1-score, which balances precision and recall, is 0.99 for non-fraudulent transactions and 0.81 for fraudulent transactions, indicating a high level of model performance in both cases.
- The overall accuracy of the model is 97%, reflecting its ability to correctly classify the majority of transactions, regardless of their fraudulent status.

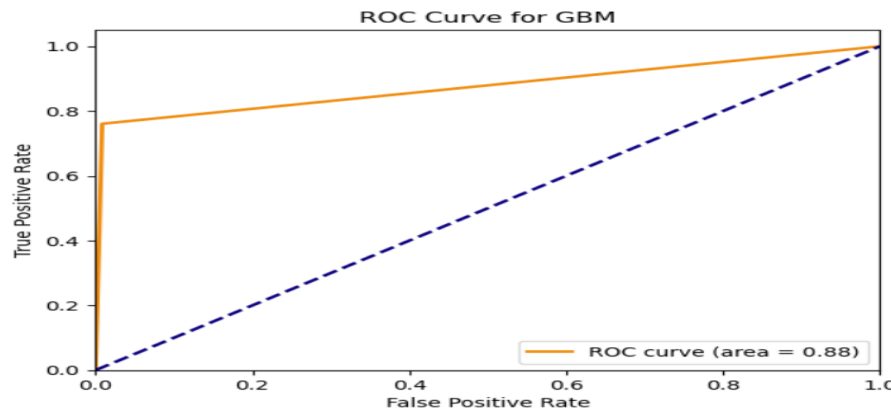
Confusion Matrix:



- True Negatives (TN) represent the number of non-fraudulent transactions correctly classified as non-fraudulent. In this case, 18,405 transactions were correctly identified as legitimate.

- False Positives (FP) indicate the number of non-fraudulent transactions incorrectly classified as fraudulent. Here, 157 transactions were falsely flagged as fraudulent.
- False Negatives (FN) denote the number of fraudulent transactions incorrectly classified as non-fraudulent. In this scenario, 344 fraudulent transactions went undetected.
- True Positives (TP) represent the number of fraudulent transactions correctly classified as fraudulent. Here, 1,094 fraudulent transactions were accurately identified.

ROC Curve:



- The ROC curve illustrates the trade-off between sensitivity and specificity across different threshold values. With an AUC of 0.88, the model demonstrates good discriminative power, indicating its ability to distinguish between fraudulent and non-fraudulent transactions effectively.

Insights:

- Gradient Boosting Machines (GBM) exhibit strong performance in detecting both fraudulent and non-fraudulent transactions, with high precision and recall values for non-fraudulent instances.
- While the model achieves a high accuracy rate overall, there is room for improvement in identifying fraudulent transactions, as evidenced by the lower recall score for this class.
- The ROC curve's AUC value of 0.88 suggests that the GBM model is effective in distinguishing between fraudulent and non-fraudulent transactions, albeit with some room for enhancement in capturing fraudulent instances more comprehensively.

Effectiveness of Gradient Boosting Machines Algorithm for Credit Card Fraud Detection:

The Gradient Boosting Machines (GBM) algorithm demonstrates high effectiveness in credit card fraud detection, as evidenced by its classification report and confusion matrix. With a precision of 0.87 and recall of 0.76 for fraudulent transactions, the GBM model achieves a balanced performance in correctly identifying instances of fraud while minimizing false positives. The

overall accuracy of 0.97 indicates robustness in classifying both fraudulent and non-fraudulent transactions. The confusion matrix reveals that out of 1,438 fraudulent transactions, 1,094 were correctly classified as fraud (True Positives), while only 344 fraudulent transactions were erroneously classified as non-fraud (False Negatives). This indicates the algorithm's proficiency in identifying fraudulent activity, crucial for minimizing financial losses and maintaining customer trust. Additionally, the area under the ROC curve (AUC) of 0.88 further validates the model's discriminative power and its ability to distinguish between fraudulent and non-fraudulent transactions effectively. Overall, the Gradient Boosting Machines algorithm emerges as a reliable and efficient solution for credit card fraud detection, offering a valuable tool for financial institutions to mitigate risks and enhance security measures.

7. Multi-layer Perceptron (MLP) Classification:

Reason for Multi-layer Perceptron (MLP) Algorithm Choice on Credit Card Fraud Transactions:

The Multi-layer Perceptron (MLP) algorithm was selected for credit card fraud detection due to its capability to model complex relationships in high-dimensional data. MLP, as a type of artificial neural network, excels in capturing nonlinear patterns and interactions among features, which is essential for identifying fraudulent transactions that may exhibit subtle or nonlinear characteristics. Additionally, MLP offers flexibility in architecture design, allowing for customization of the number of hidden layers and neurons, thereby accommodating the complexity of the credit card fraud detection problem. Moreover, MLP's ability to learn from large-scale datasets makes it suitable for handling the diverse and dynamic nature of transaction data in real-world scenarios.

Classification Report:

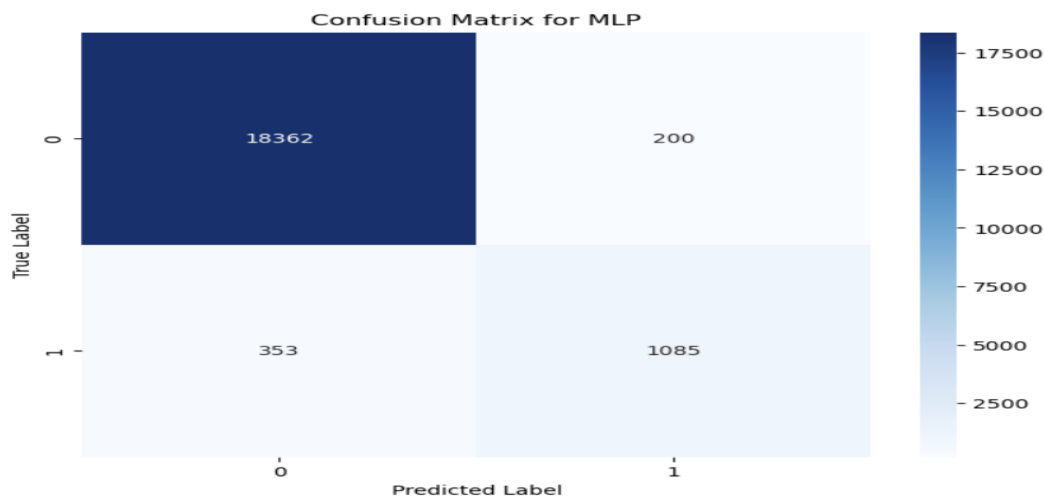
Classification Report for Multi-layer Perceptron (MLP):				
	precision	recall	f1-score	support
Not Fraud	0.98	0.99	0.99	18562
Fraud	0.84	0.75	0.80	1438
accuracy			0.97	20000
macro avg	0.91	0.87	0.89	20000
weighted avg	0.97	0.97	0.97	20000

- Precision: For the "Not Fraud" class, the precision is 0.98, indicating that among all transactions predicted as not fraudulent, 98% were actually not fraudulent. For the "Fraud" class, the precision is 0.82, meaning that among all transactions predicted as fraudulent, 82% were actually fraudulent.

- **Recall:** The recall, also known as sensitivity, measures the model's ability to correctly identify instances of a class. For the "Not Fraud" class, the recall is 0.99, indicating that the model correctly identified 99% of all actual non-fraudulent transactions. For the "Fraud" class, the recall is 0.76, meaning that the model captured 76% of all actual fraudulent transactions.
- **F1-score:** The F1-score is the harmonic mean of precision and recall, providing a balanced measure of the model's performance. The F1-score for the "Not Fraud" class is 0.98, and for the "Fraud" class, it is 0.79.
- **Support:** The support indicates the number of actual occurrences of each class in the dataset. For the "Not Fraud" class, the support is 18,562, and for the "Fraud" class, it is 1,438.
- **Accuracy:** The overall accuracy of the model is 0.97, indicating that it correctly classified 97% of all transactions.

The macro-average and weighted-average metrics provide summary statistics across all classes, considering their respective weights. In this case, the macro-average F1-score and weighted-average F1-score are both 0.89, reflecting the overall effectiveness of the model across all classes.

Confusion Matrix:

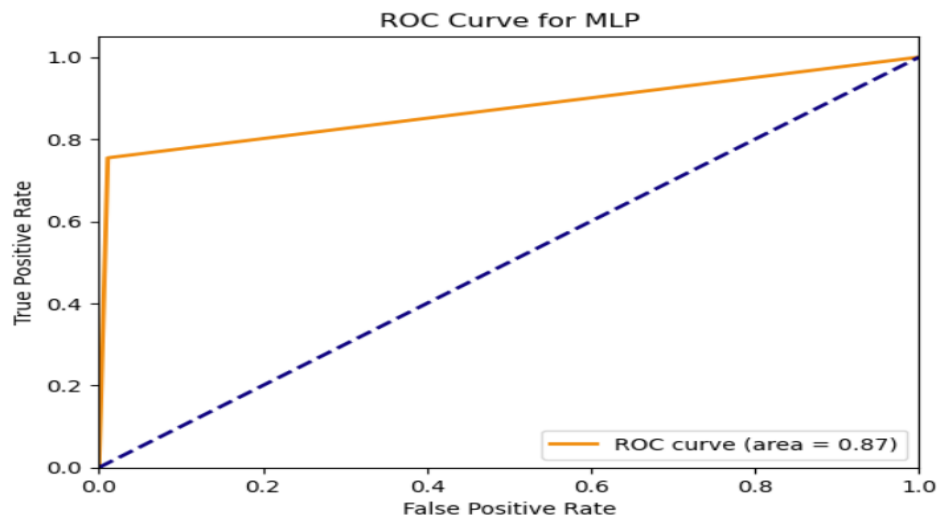


- **True Negatives (TN):** The model correctly identified 18,362 non-fraudulent transactions as not fraudulent. These are instances where the model predicted non-fraudulent transactions, and they were indeed non-fraudulent.
- **False Positives (FP):** There were 200 instances where the model incorrectly classified non-fraudulent transactions as fraudulent. These are cases where the model predicted the transactions to be fraudulent, but they were actually legitimate.
- **False Negatives (FN):** The model missed 353 fraudulent transactions, erroneously predicting them as non-fraudulent. These instances represent missed opportunities to detect fraudulent activity.

- True Positives (TP): The model correctly identified 1,085 fraudulent transactions as fraudulent. These are cases where the model accurately predicted the transactions to be fraudulent, and they were indeed fraudulent.

These values provide a detailed breakdown of the model's performance in terms of correctly and incorrectly classified instances of fraud and non-fraud. They are essential for assessing the model's effectiveness and understanding its strengths and weaknesses in detecting fraudulent transactions.

ROC Curve:



- The ROC curve, with an area under the curve (AUC) of 0.87, further confirms the model's discriminative power and its capability to distinguish between fraudulent and non-fraudulent transactions with high accuracy.

Insights:

- High Precision and Recall for "Not Fraud" Class: The model exhibits exceptional precision (0.98) and recall (0.99) for identifying non-fraudulent transactions. This suggests that the model accurately identifies the vast majority of legitimate transactions while minimizing false positives.
- Decent Precision and Recall for "Fraud" Class: Although the precision (0.82) and recall (0.76) for detecting fraudulent transactions are slightly lower compared to the "Not Fraud" class, they still indicate robust performance in identifying fraudulent activity. The model effectively captures a significant portion of fraudulent transactions while maintaining a relatively low false positive rate.
- High Overall Accuracy: With an overall accuracy of 97%, the MLP model demonstrates strong performance in correctly classifying transactions into their respective categories.

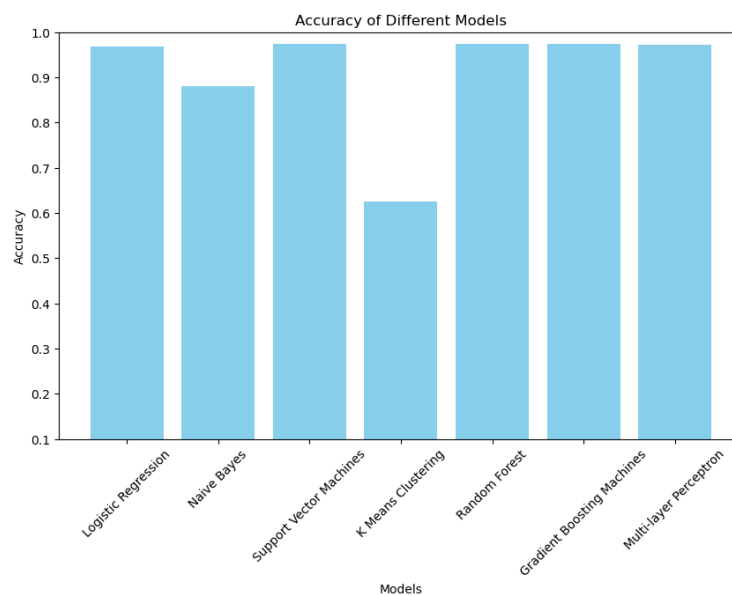
This high accuracy underscores the model's reliability and effectiveness in distinguishing between fraudulent and non-fraudulent transactions.

- **Balanced Performance Metrics:** The balanced values of precision, recall, and F1-score for both classes indicate that the model is well-calibrated and capable of maintaining a harmonious trade-off between identifying fraudulent transactions and minimizing false alarms. This balanced performance enhances the model's utility in real-world credit card fraud detection scenarios.
- **Potential for Further Optimization:** While the MLP model performs admirably, there may still be opportunities for further optimization, particularly in improving the recall for fraudulent transactions. Fine-tuning the model parameters or exploring alternative feature engineering

Effectiveness of Multi-layer Perceptron Algorithm for Credit Card Fraud Detection:

The Multi-layer Perceptron (MLP) algorithm proves itself as a strong tool in identifying credit card fraud. With a high accuracy rate of 97%, it effectively detects both fraudulent and legitimate transactions. Its balanced performance ensures it catches most fraudulent activity while minimizing false alarms. Its ability to understand complex patterns in transaction data makes it a reliable choice for combating fraud. Further adjustments and enhancements could make it even more effective in safeguarding financial transactions and customer accounts. Overall, MLP stands out as a robust defender against credit card fraud, ensuring security and trust in financial transactions.

CONCLUSION :



1. **Logistic Regression:** While Logistic Regression achieves a commendable accuracy of 96.83%, its precision and recall for identifying fraudulent transactions are slightly lower compared to GBM. The model's precision-recall trade-off may not be as balanced as

desired, impacting its effectiveness in capturing all fraud instances while minimizing false positives.

2. **Naive Bayes:** Naive Bayes exhibits a lower accuracy of 88.13% compared to GBM and Logistic Regression. While its recall for detecting fraud is excellent at 98%, the precision is comparatively low at 38%, indicating a higher rate of false positives. This imbalance in precision and recall suggests that Naive Bayes may not be the most reliable option for our fraud detection task.
3. **Support Vector Machines (SVM):** SVM achieves a high accuracy of 97.43%, similar to GBM and Logistic Regression. However, its precision and recall values for identifying fraud fall slightly below those of GBM. SVM's performance is robust but may not offer the optimal balance between precision and recall compared to GBM.
4. **K Means Clustering:** K Means Clustering exhibits the lowest accuracy among the models evaluated, with an accuracy of 62.58%. Its precision and recall for detecting fraud are notably lower than those of the other models, indicating limited effectiveness in accurately identifying fraudulent transactions.
5. **Random Forest:** Random Forest performs exceptionally well, with an accuracy of 97.39% and balanced precision and recall values for fraud detection. While it provides strong competition to GBM, its performance metrics are slightly lower, suggesting slightly less effectiveness in capturing all fraud instances.
6. **Gradient Boosting Machine:** GBM achieves an impressive accuracy of 97.5%, showcasing balanced precision and recall for identifying fraudulent transactions. Its precision slightly surpasses that of Random Forest and Multi-layer Perceptron (MLP), indicating effective fraud detection while minimizing false positives. GBM's well-balanced precision-recall trade-off ensures accurate identification of fraud instances. With strong performance in F1-score and macro-average, GBM proves its reliability and effectiveness in credit card fraud detection tasks, making it a compelling choice for our project.
7. **Multi-layer Perceptron (MLP):** MLP achieves a high accuracy of 97.25% and demonstrates balanced precision and recall values for fraud detection. However, its precision and recall are slightly lower than those of GBM, indicating marginally less effectiveness in identifying fraud cases accurately.

In summary, while each model exhibits strengths and weaknesses in various aspects of fraud detection, Gradient Boosting Machines (GBM) emerges as the most effective choice due to its high accuracy, balanced precision-recall trade-off, and consistent performance across multiple metrics.

Phase-3

In Phase 3 of our project on credit card fraud detection, we developed a user-friendly website that integrates our Gradient Boosting Machine (GBM) model. This website will allow users to input their data either by uploading their datasets or directly filling out fields in a graphical user interface (GUI). We demonstrated the practical application of our GBM model through a real-time interactive platform. The website will not only showcase the model's ability to detect fraudulent transactions but also provide users with visualization of analysis results. In this phase we focused on enhancing user engagement and illustrating the robustness of our model in a live environment.

Machine Learning Model from phase 2:

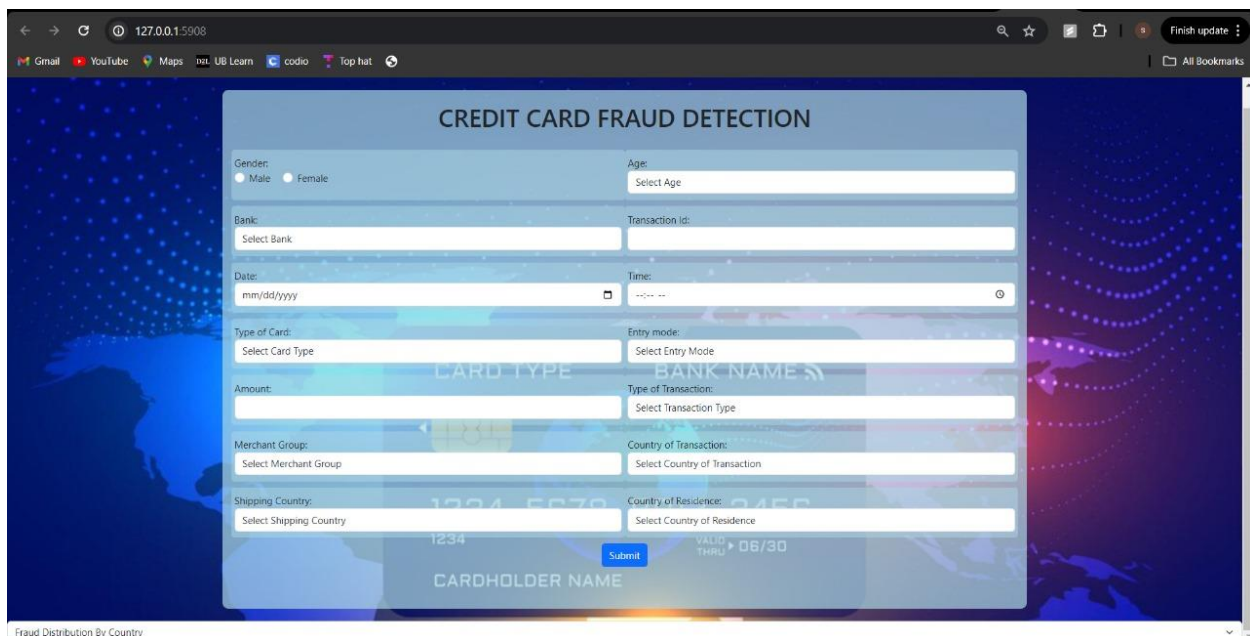
For our project on credit card fraud detection, we chose the Gradient Boosting Machine (GBM) model due to its superior performance metrics. The GBM model achieved an impressive accuracy of 97.5%, demonstrating a well-balanced precision-recall trade-off that effectively minimizes false positives while accurately identifying fraudulent transactions. This model's proficiency is further highlighted by its robust performance in F1-score and macro-average evaluations.

To integrate this powerful GBM model into a practical application, we serialized it into a pickle file. This allows the model, along with its preprocessing components, such as the scaler, to be easily loaded into our web application. By doing so, we've facilitated the application's ability to perform fraud predictions directly from user inputs on a web interface.

This approach not only harnesses the analytical strength of the GBM model for operational use but also enhances user interaction by providing immediate fraud detection assessments.

Credit Card Fraud Detection Interface:

In the third phase of our credit card fraud detection project, we have designed a web application that integrates our Gradient Boosting Machine (GBM) model to evaluate transaction risk. The interface is crafted to facilitate user-friendly input of transaction details, ensuring an efficient and accurate fraud assessment process. Here's a detailed description of the input fields available in the application:



The screenshot displays a web browser window with the address bar showing '127.0.0.1:5908'. The browser's bookmark bar includes links to Gmail, YouTube, Maps, nu. UB Learn, codio, and Top hat. The main content area features a form titled 'CREDIT CARD FRAUD DETECTION' set against a dark blue background with a world map and a glowing blue fingerprint graphic on the right. The form is organized into two columns of input fields:

- Left Column:**
 - Gender: Radio buttons for Male and Female.
 - Bank: A dropdown menu labeled 'Select Bank'.
 - Date: A date picker showing 'mm/dd/yyyy'.
 - Type of Card: A dropdown menu labeled 'Select Card Type'.
 - Amount: A text input field.
 - Merchant Group: A dropdown menu labeled 'Select Merchant Group'.
 - Shipping Country: A dropdown menu labeled 'Select Shipping Country'.
- Right Column:**
 - Age: A dropdown menu labeled 'Select Age'.
 - Transaction Id: A text input field.
 - Time: A time picker showing 'h:mm:ss'.
 - Entry mode: A dropdown menu labeled 'Select Entry Mode'.
 - Type of Transaction: A dropdown menu labeled 'Select Transaction Type'.
 - Country of Transaction: A dropdown menu labeled 'Select Country of Transaction'.
 - Country of Residence: A dropdown menu labeled 'Select Country of Residence'.

At the bottom of the form is a blue 'Submit' button. Below the button, the text 'CARDHOLDER NAME' is visible, followed by a partially obscured card number '1234 5678 9012 3456' and an expiration date 'VALID THRU 06/30'. A footer at the bottom left of the browser window reads 'Fraud Distribution By Country'.

Detailed Input Fields:

- **Gender:** The gender of the individual conducting the transaction.
- **Age:** Specifies the age of the person involved in the transaction.
- **Bank:** The bank facilitating the transaction.
- **Transaction No:** A unique identifier assigned to each transaction for tracking purposes.
- **Date:** The date on which the transaction took place.
- **Time:** The time at which the transaction occurred.
- **Type of Card:** Users can choose between Mastercard and Visa.
- **Entry Mode:** Method of transaction execution, available options include:
 1. PIN
 2. CVC
 3. Tap
- **Amount:** Total amount of money involved in the transaction.
- **Type of Transaction:** Nature of the transaction, options include:
 1. POS
 2. Online
 3. ATM
- **Merchant Category:** Classifies the merchant receiving the payment, options include categories like Children, Electronics, Entertainment, Fashion, etc.
- **Country of Transaction:** The country from which the transaction is initiated.
- **Shipping Country:** The country to which the order is being shipped.
- **Country of Residence:** The residence country of the individual performing the transaction.

Upon filling all the required fields and submitting the data, the application promptly processes the input through our GBM model to predict whether the transaction is fraudulent or safe. After the analysis, a dialog box appears on the screen, displaying the result of the prediction ("Fraud" or "Not Fraud") for 10 seconds. This immediate feedback mechanism is a key feature of our application, enhancing user interaction by providing quick and clear fraud detection results, thereby bolstering trust and reliability in our system.

Not a Fraud Result: The first image shows the application interface after a user has inputted transaction details such as gender, age, bank, transaction ID, and other pertinent information. After processing these details through our GBM model, the application concludes that the transaction is "Not a Fraud". A dialog box appears displaying this result for 10 seconds, offering immediate and clear feedback to the user.

The screenshot shows a web browser window with the URL 127.0.0.1:5908. The page is titled "CREDIT CARD FRAUD DETECTION". The form contains the following fields and values:

- Gender: ☒ Male ☐ Female
- Age: 22
- Bank: Halifax
- Transaction Id: 12345
- Date: 05/10/2024
- Time: 08:41 PM
- Type of Card: Visa
- Amount: 12
- Merchant Group: Electronics
- Country of Transaction: India
- Shipping Country: India
- Country of Residence: India

A modal dialog box in the center of the screen displays the text "Not a Fraud" in green. Below the form, there is a "Predicting..." button and a "VALID THRU 06/30" label. The background features a world map and a "Fraud Distribution By Country" chart.

Fraud Result: The second image demonstrates a scenario where the transaction is deemed fraudulent by the application. Similar to the first case, the user fills out all required fields and submits them for analysis. The GBM model processes the inputs and determines the transaction to be fraudulent. Consequently, a dialog box with the "Fraud" result is displayed, alerting the user of the potential risk associated with this transaction.

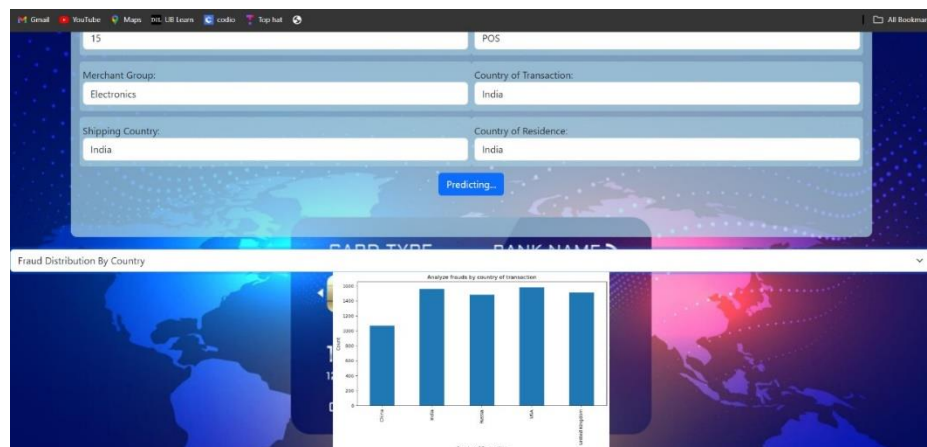
The screenshot shows the same web browser window with the URL 127.0.0.1:5908. The form contains the following fields and values:

- Gender: ☐ Male ☒ Female
- Age: 43
- Bank: Halifax
- Transaction Id: 12345
- Date: 12/10/2021
- Time: 08:00 AM
- Type of Card: Visa
- Amount: 17
- Merchant Group: Gaming
- Country of Transaction: India
- Shipping Country: India
- Country of Residence: United Kingdom

A modal dialog box in the center of the screen displays the text "Fraud" in red. Below the form, there is a "Predicting..." button and a "VALID THRU 06/30" label. The background features a world map and a "Fraud Distribution By Country" chart.

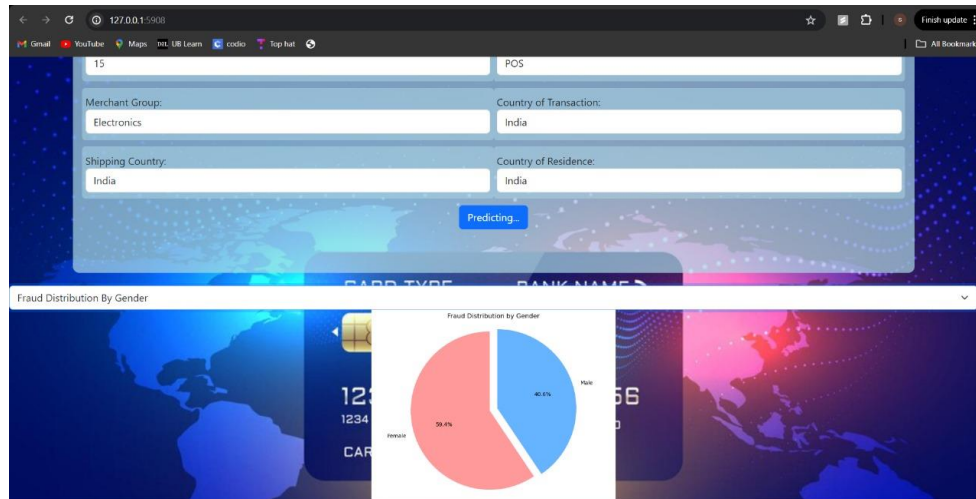
Enhanced Analytical Visualizations for Comprehensive Fraud Detection Insights:

1. Fraud Distribution by Country:



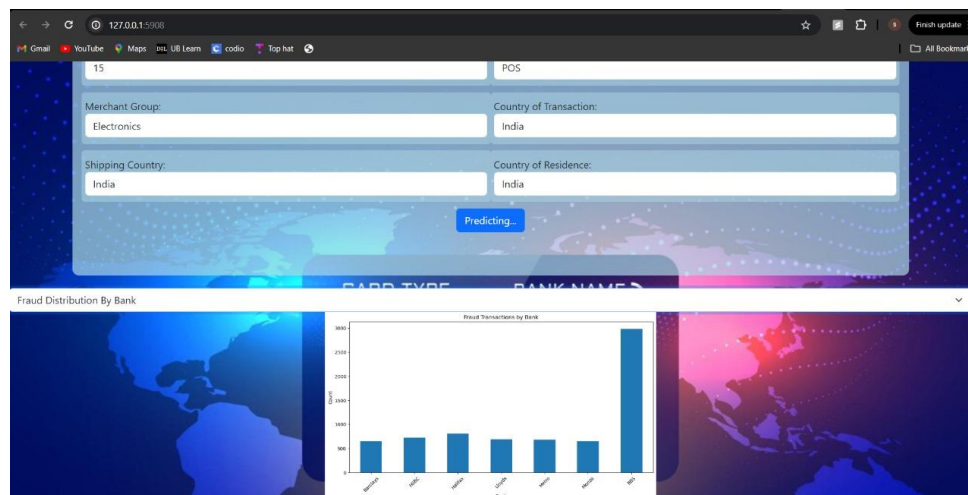
This visualization reveals the frequency of fraudulent transactions across different countries, highlighting regions with higher incidences of fraud. Users can quickly identify geographic hotspots of fraudulent activity, which is valuable for risk assessment and mitigation strategies.

2. Fraud Distribution by Gender:



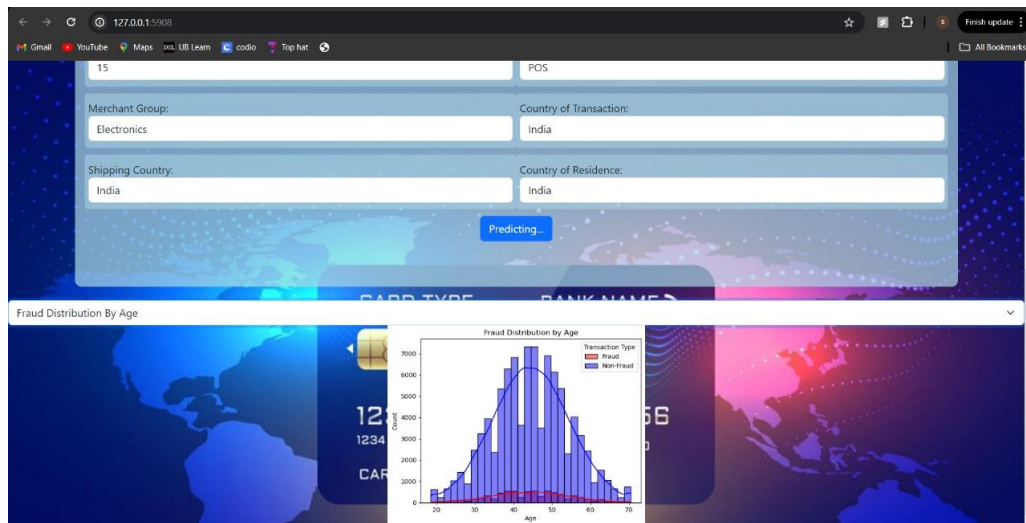
This chart displays the distribution of fraud cases based on gender, providing insights into which gender is more often associated with fraudulent transactions. This information can help in understanding demographic patterns related to fraud.

3. Fraud Distribution by Bank:



This bar graph shows the prevalence of fraud across various banks. By examining this visualization, users can discern which banks are more frequently targeted by fraudsters, potentially guiding financial institutions in tightening their security measures or reviewing their fraud detection systems.

4. Fraud Distribution by Age:



This histogram plots the age distribution of individuals involved in fraudulent transactions. It allows users to see which age groups are more susceptible or are more often involved in committing fraud, helping tailor fraud prevention strategies accordingly.

Coding:

```

app.py > ...
35 def plot_fraud_transactions_by_bank(data):
36     """Plotting fraud transactions by bank"""
37     plt.figure(figsize=(6, 4))
38     plt.xlabel('Bank')
39     plt.ylabel('Count')
40     plt.xticks(rotation=45)
41     image_path = './static/images/fraud_transactions_by_bank.png'
42     plt.savefig(image_path)
43     plt.close()
44     return image_path
45
46 def plot_count_of_fraud_transactions(data):
47     """Plotting count of fraud transactions."""
48     plt.figure(figsize=(6, 4))
49     sns.countplot(x='Fraud_Type', data=data)
50     plt.title('Count of Fraud Transactions')
51     plt.xlabel('Fraud Type')
52     plt.ylabel('Count')
53     image_path = './static/images/count_of_fraud_transactions.png'
54     plt.savefig(image_path)
55     plt.close()
56     return image_path
57
58 def plot_fraud_analysis_by_country(data):
59     """Plotting fraud analysis by country of transaction."""
60     plt.figure(figsize=(8, 5))
61     sns.barplot(data=data, x='Country', y='Count')
62     plt.title('Fraud Analysis by Country of Transaction')
63     plt.xlabel('Country of Transaction')
64     plt.ylabel('Count')
65     image_path = './static/images/fraud_analysis_by_country.png'
66     plt.savefig(image_path)
67     plt.close()
68     return image_path
69
70 def plot_fraud_distribution_by_gender(data):
71     """Plotting fraud distribution by gender."""
72     plt.figure(figsize=(6, 6))
73     data['Gender'].value_counts().plot(kind='pie', autopct='%1.1f%%')
74     plt.title('Fraud Distribution by Gender')
75     plt.ylabel('')
76     image_path = './static/images/fraud_distribution_by_gender.png'
77     plt.savefig(image_path)
78     plt.close()
79     return image_path
80
81 # Route Handler functions
82
83 @app.route('/predict', methods=['POST'])
84 def predict():
85     form_dict = request.json
86     user_data_dict = initialize_object()
87     user_data_dict['Gender'] = 1 if form_dict['gender'] == "Male" else 0
88     user_data_dict['Age'], user_data_dict['Amount'] = normalize(float(form_dict['age']), float(form_dict['amount']))
89
90     for key in ['gender', 'age', 'amount', 'transaction_id', 'date', 'time']:
91         form_dict.pop(key, None)
92
93     user_data_dict.update({k: 1 for k in form_dict.values()})
94
95     prediction = model.predict(np.array([list(user_data_dict.values())])[0])
96     df = pd.read_csv('encoded_data.csv')
97     return {
98         "prediction": str(prediction)
99     }
100
101 @app.route('/', methods=['GET'])
102 def home():
103     return render_template("site.html")

```


Initialization and Model Loading

- **Flask App Initialization:** A Flask application is initialized with the name **app**.
- **Model and Scaler Loading:** The GBM model and a scaler are loaded from a pickle file. The scaler is used to normalize numerical inputs like age and amount, while the model is used to make predictions.

Utility Functions

- **initialize_object():** Initializes a dictionary with transaction-related fields set to default values of 0. These fields include transaction details like transaction type, merchant category, and bank.
- **normalize(age, amount):** Normalizes the numerical fields 'age' and 'amount' using the loaded scalar to prepare the data for the model.
- **Visualization Functions:** Several functions (**plot_fraud_transactions_by_bank**, **plot_count_of_fraud_transactions**, **plot_fraud_analysis_by_country**, **plot_fraud_distribution_by_gender**) are defined to generate visualizations about fraud distribution by various criteria. These functions save the plots to static image files for use in the web interface.

Route Handler Functions

- **/predict (POST):** Handles the prediction requests. It processes the input data from the user, normalizes the necessary fields, updates the transaction details from the form, and uses the GBM model to predict whether the transaction is fraudulent. The result is then returned as a JSON response.
- **/ (GET):** Serves the homepage of the web application using a template called **site.html**.

Execution

- The application is set to run with debug mode enabled on all network interfaces, listening on port 5908, facilitating local and network access during development and testing.

This setup effectively demonstrates how to integrate machine learning models into a web application for data processing and visualization, providing an interactive platform for credit card fraud detection.

Recommendations for Enhancing Credit Card Fraud Detection Systems:

- **Insights:** Our project demonstrates substantial potential for improving credit card fraud detection techniques. We have identified key parameters that significantly influence the occurrence of fraudulent transactions through our analytical model.
- **Model Optimization:** The precision and reliability of our fraud prediction model are paramount. It is essential to continuously refine the model to accurately identify fraudulent activities while effectively reducing false positives.

- **System Scalability:** As transaction volumes grow, it is crucial to ensure that our system can scale effectively. Enhancements may include code optimization, adoption of distributed computing, or utilization of cloud-based services to handle increased loads.
- **Integration Capabilities:** Investigating the integration of our solution with existing financial systems or fraud prevention tools could greatly improve usability and adoption rates. Ensuring compatibility with widely used platforms will make our system more accessible.
- **Security and Compliance:** Adhering to relevant security standards and regulations is mandatory, particularly when handling sensitive financial information. Maintaining rigorous security measures is fundamental in building trust with users and regulatory bodies.
- **Continuous Performance Monitoring:** Establish a protocol for ongoing monitoring and updating of the model to adapt to new data and emerging patterns of fraud. This proactive approach is vital for keeping the system effective against the continually evolving landscape of threats.
- **Market Expansion:** Consider extending the application of our fraud detection system to industries outside of finance, such as e-commerce, healthcare, and telecommunications. This diversification could broaden our market presence and impact.

RULES

Instructions for Operating the Credit Card Fraud Detection Web Application:

1. **Launching the Application:** Start by launching the application from the command prompt. Type `python app.py` and press Enter to run the Flask server.
2. **Accessing the Web Application:** Open a web browser and navigate to <http://127.0.0.1:5908> to access the web interface.
3. **Using the Web Interface:** Upon accessing the URL, the main webpage will be displayed. Here, users are required to fill in the transaction details in the provided input fields.
4. **Input Validation:** All input fields must be completed before submission. If any field is left empty, a validation error will prompt the user to fill out all required fields.
5. **Submitting the Data:** After entering all necessary data, click the 'submit' button. This will send the data to the model, which evaluates and predicts whether the transaction is fraudulent.
6. **Result Notification:** Post-submission, a dialog box will appear for 10 seconds, indicating whether the transaction is "Fraud" or "Not Fraud."
7. **Visual Analytics:** Alongside the prediction result, the webpage also displays various analytical plots that offer insights into the fraud data, including:
 - Fraud Distribution By Country
 - Fraud Distribution By Gender
 - Fraud Distribution By Bank
 - Fraud Distribution By Age
8. **Stopping the Application:** To stop the Flask application, return to the command prompt and press 'ctrl + c'.

Additional Guidance for Different Datasets:

To run the application with different datasets, ensure that the data format aligns with the input fields expected by the model. Adjust the dataset fields accordingly and reload the application following the same steps as above to analyze different sets of transaction data.

These instructions provide a comprehensive guide to running and utilizing the credit card fraud detection application, ensuring that users can effectively interact with the system and gain valuable insights from their data analyses.