Analysis of Credit Card Fraud Dataset

Author: Mehrdad Baradaran

Abstract

Fraud detection is a critical aspect of financial security in today's technologically advanced world. This assignment delves into the realm of data science to develop models for predicting fraudulent credit card transactions. The dataset used for this task spans from January 1, 2019, to December 31, 2020, containing a mix of legitimate and fraudulent transactions from a diverse set of customers and merchants.

The focus of this assignment is on supervised learning, specifically discrete variable prediction, employing various machine learning algorithms. The chosen models include Logistic Regression, Support Vector Machine (SVM), K-nearest neighbor (KNN), Decision Tree, Random Forest, and Naive Bayes classifiers. The goal is to build models that accurately classify transactions as either fraudulent or legitimate.

The dataset will undergo preprocessing steps such as feature engineering and selection before training the models. The performance of each model will be assessed on both the training and test sets, with metrics such as accuracy, AUC curve, confusion matrix, F1 score, and decision boundary being reported. A critical aspect of this assignment is the implementation of a sklearn-based pipeline to ensure reproducibility and scalability.

Introduction

Fraudulent activities in credit card transactions pose significant threats to financial institutions and their customers. As fraudsters evolve their tactics, traditional methods of detection become less effective, necessitating advanced data science approaches. This assignment focuses on leveraging machine learning algorithms to create robust fraud detection models.

The dataset used for this task originates from a simulated credit card transaction environment spanning two years. It includes diverse transaction profiles, encompassing various customers, merchants, and transaction categories. The goal is to predict whether a given transaction is fraudulent or not.

The chosen machine learning models cover a spectrum of algorithms, each with its unique strengths in handling classification tasks. Logistic Regression, Support Vector Machine, K-nearest neighbor, Decision Tree, Random Forest, and Naive Bayes classifiers will be employed and evaluated comprehensively.

Preprocessing steps are crucial to enhance the model's predictive capabilities. Feature engineering and selection will be applied to extract relevant information and optimize model performance. The assessment of model accuracy on both training and test sets will provide insights into their generalization capabilities.

The implementation of a sklearn-based pipeline ensures a systematic and reproducible approach to model development. The assignment not only aims for accurate predictions but also emphasizes the importance of understanding model behavior through visualization and interpretation metrics. The subsequent sections will delve into the methodology, results, and discussions, shedding light on the efficacy of each classification algorithm in tackling credit card fraud detection.

Train Data Information:

The training dataset contains a total of 1,296,675 entries and 23 columns. Here is an overview of the data types and non-null counts for each column:

1. **Data Types:**
    - Integer64: 6 columns
    - Float64: 5 columns
    - Object (String): 12 columns
2. **Key Columns:**
    - Unnamed: 0: Index column with integer values.
    - trans_date_trans_time: Transaction date and time represented as an object.
    - cc_num: Credit card number represented as an integer.
    - merchant: Merchant name represented as a string.
    - amt: Amount of transaction represented as a float.
    - is_fraud: Binary indicator for fraud, represented as an integer.
3. **Memory Usage:**
    - The dataset consumes approximately 227.5 MB of memory.

Test Data Information:

The test dataset comprises 555,719 entries and shares the same structure with the training dataset, containing 23 columns. Here is a summary of the data types and non-null counts:

1. **Data Types:**
    - Integer64: 6 columns
    - Float64: 5 columns
    - Object (String): 12 columns
2. **Key Columns:**
    - Unnamed: 0: Index column with integer values.
    - trans_date_trans_time: Transaction date and time represented as an object.
    - cc_num: Credit card number represented as an integer.
    - merchant: Merchant name represented as a string.
    - amt: Amount of transaction represented as a float.
    - is_fraud: Binary indicator for fraud, represented as an integer.

Check for missing values in both datasets:

```
Missing Values in Train Data:
Unnamed: 0              0
trans_date_trans_time   0
cc_num                  0
merchant                0
category                0
amt                     0
first                   0
last                    0
gender                  0
street                  0
city                    0
state                   0
zip                     0
lat                     0
long                    0
city_pop                0
job                     0
dob                     0
trans_num               0
unix_time               0
merch_lat               0
merch_long              0
is_fraud                0
dtype: int64
```

It seems that there are no missing values in both the training and test datasets. This is great news, as having complete data ensures a solid foundation for building and training machine learning models.

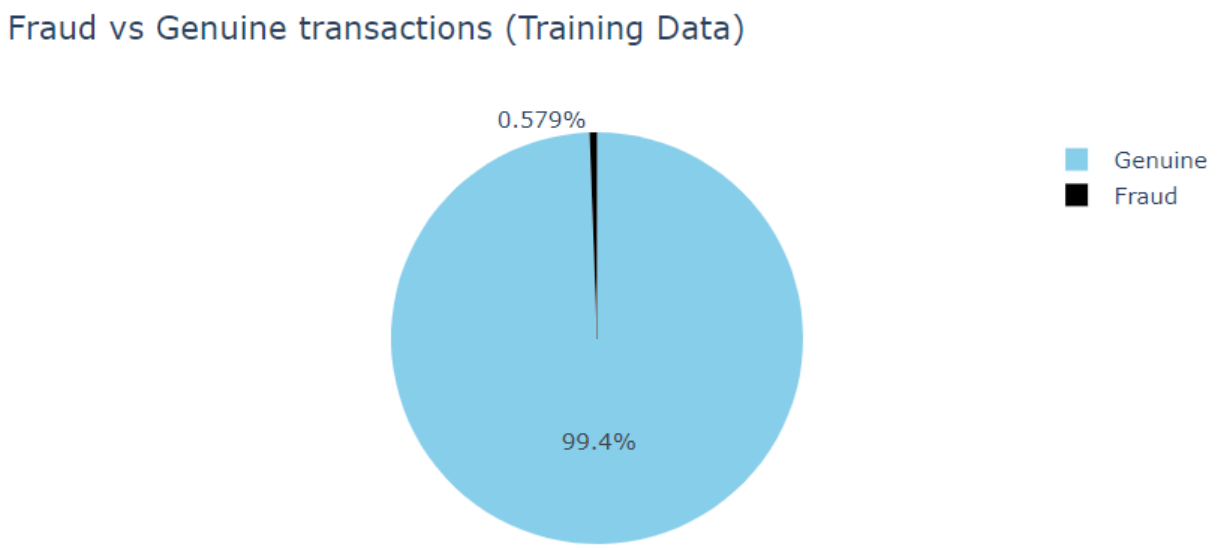
1. **Training Set Fraud Percentage (0.579%):**
    - In the training dataset, the fraud class constitutes an extremely small percentage, approximately 0.579%.
    - This indicates an exceptionally imbalanced class distribution, where instances of fraud are exceedingly rare compared to non-fraudulent transactions.
2. **Test Set Fraud Percentage (0.386%):**
    - In the test dataset, the percentage of transactions labeled as fraud is even smaller, at approximately 0.386%.
    - The test set exhibits an even lower frequency of fraud instances compared to the training set.

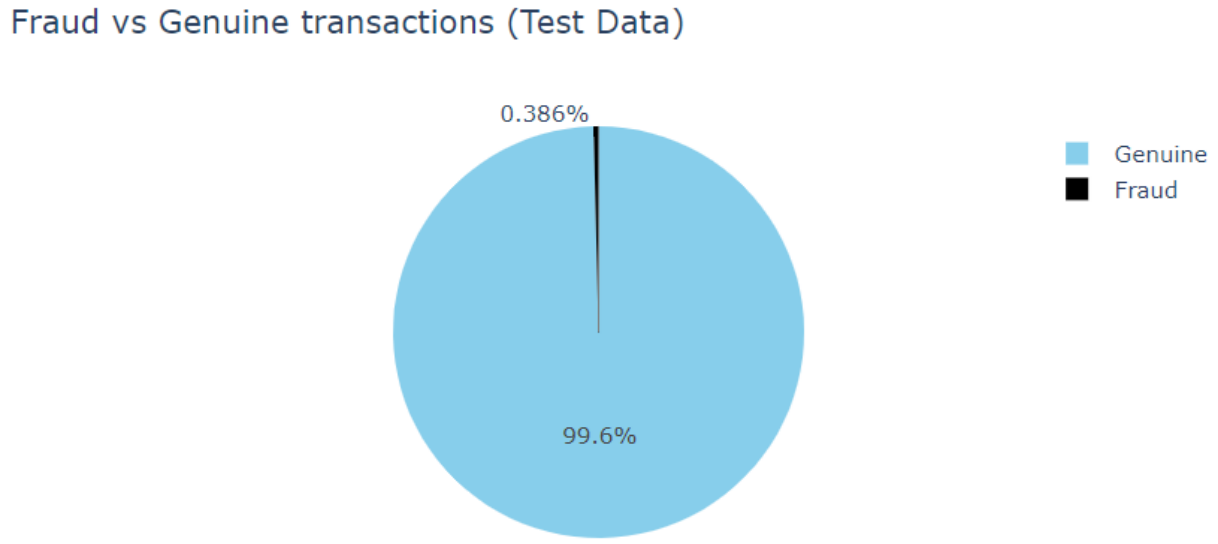
Interpretation:

- The minuscule percentages underscore the rarity of fraud instances in both the training and test sets.
- The severe class imbalance highlights the need for specialized evaluation metrics and model tuning.

Recommendations:

- Given the extremely low percentage of fraud instances, careful consideration of evaluation metrics like precision, recall, F1-score, and ROC-AUC is essential.
- Experiment with various techniques to address class imbalance during model training, such as oversampling, undersampling, or adjusting class weights.
- Fine-tune the model parameters and features, taking into account the challenges posed by the highly imbalanced nature of the data.



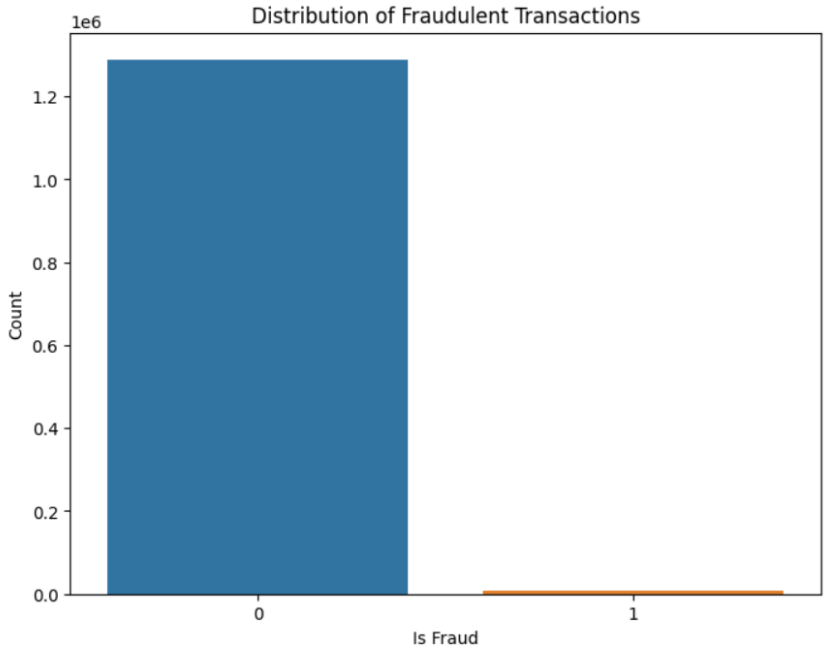


Caution in Using Accuracy Score for Imbalanced Datasets:

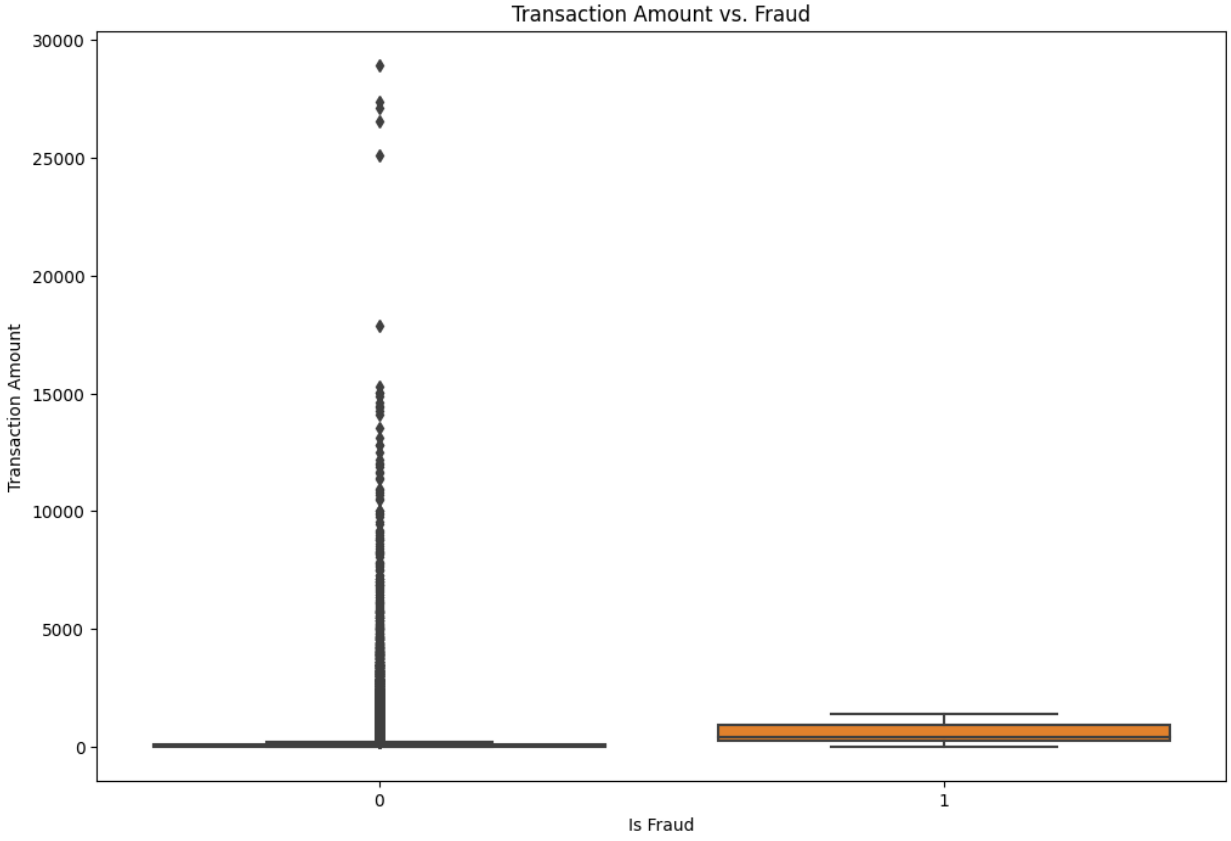
Using accuracy score as a metric for evaluating model performance in imbalanced datasets is not recommended. In our dataset, there is a substantial class imbalance, with 99.42% of transactions labeled as Genuine and only 0.579% (7,506) labeled as fraud. If we were to blindly guess the majority class (Genuine), we would achieve an accuracy of 99.42%. However, relying solely on accuracy can be misleading in such imbalanced scenarios.

Exploratory Data Analysis (EDA)

Visualize the distribution of the target variable (fraudulent or not)



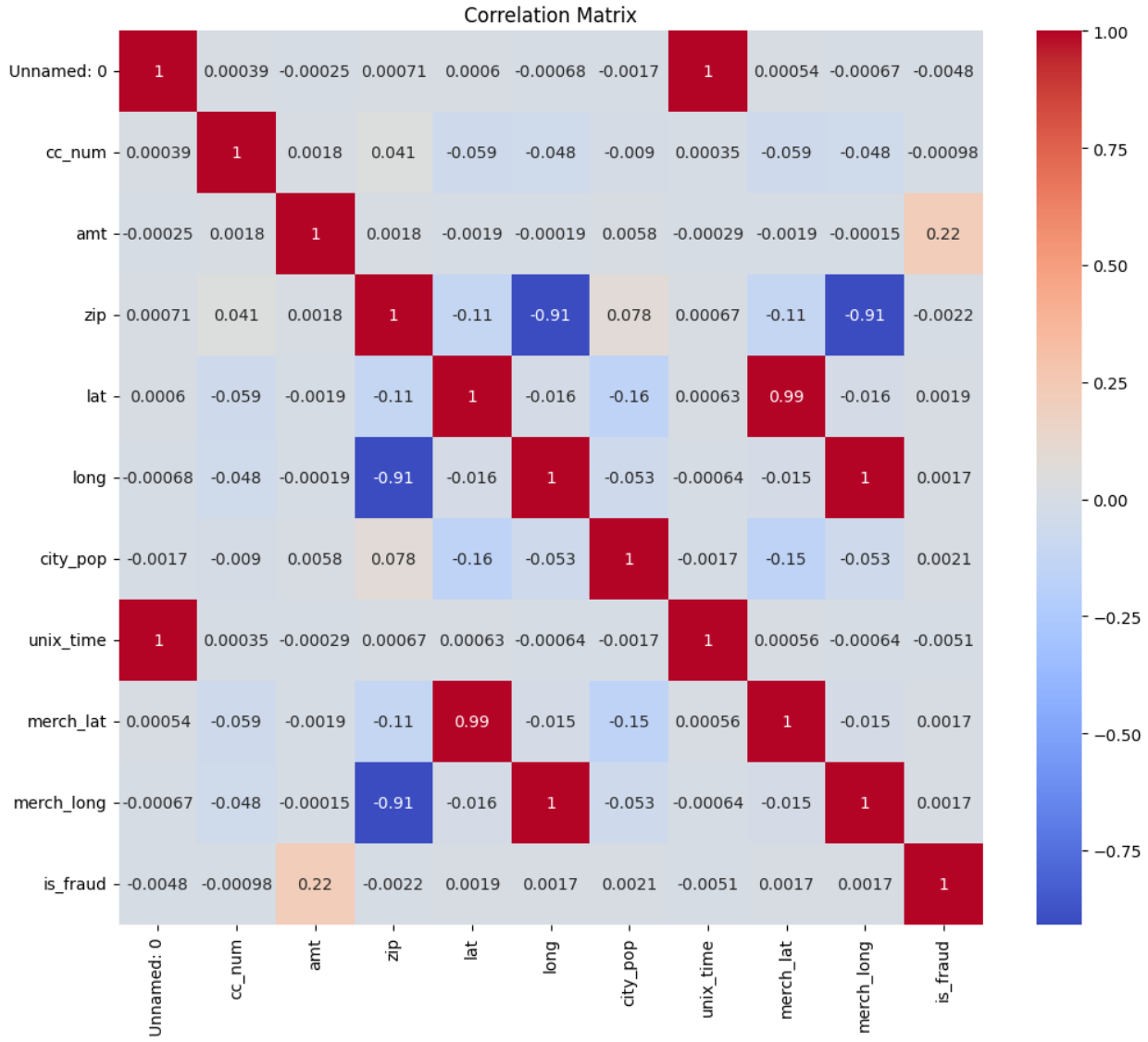
Explore the distribution of transaction amounts by fraud status



Our analysis of the dataset reveals a substantial class imbalance in the distribution of transactions:

- **Count Plot:** Approximately 99% of transactions are labeled as not fraudulent (Class 0), while only a minimal fraction is marked as fraudulent (Class 1). This skewed distribution is visually evident in the count plot.
- **Box Plot:** The box plot illustrates the spread of transaction amounts for each class. Genuine transactions (Class 0) show a typical range, whereas fraudulent transactions (Class 1) exhibit a more varied distribution, albeit with fewer instances.

Exclude non-numeric columns from the correlation matrix

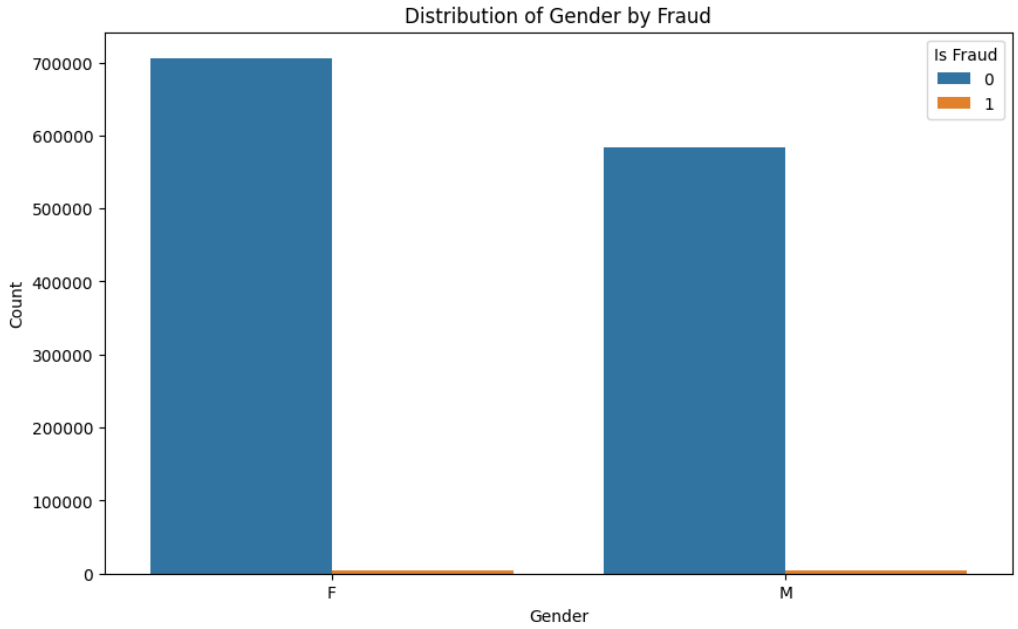


Our examination of the numerical features within the dataset reveals notable correlations between specific pairs:

- **Long and Merch_Long (Correlation > 0.9):** The longitude coordinates of transactions exhibit a strong positive correlation with the corresponding merchant longitude. This implies a spatial relationship, suggesting transactions tend to occur in specific geographical clusters.
- **Lat and Merch_Lat (Correlation > 0.9):** Similar to the longitude coordinates, the latitude coordinates of transactions and merchant latitude display a significant positive correlation. This reinforces the spatial association observed in the longitude coordinates.
- **Zip and Merch_Long (Correlation < -0.9):** Surprisingly, there is a strong negative correlation between the zip code and merchant longitude. This may indicate that certain regions represented by zip codes align with distinct merchant longitudes, revealing a potential geographical pattern.

And so on.

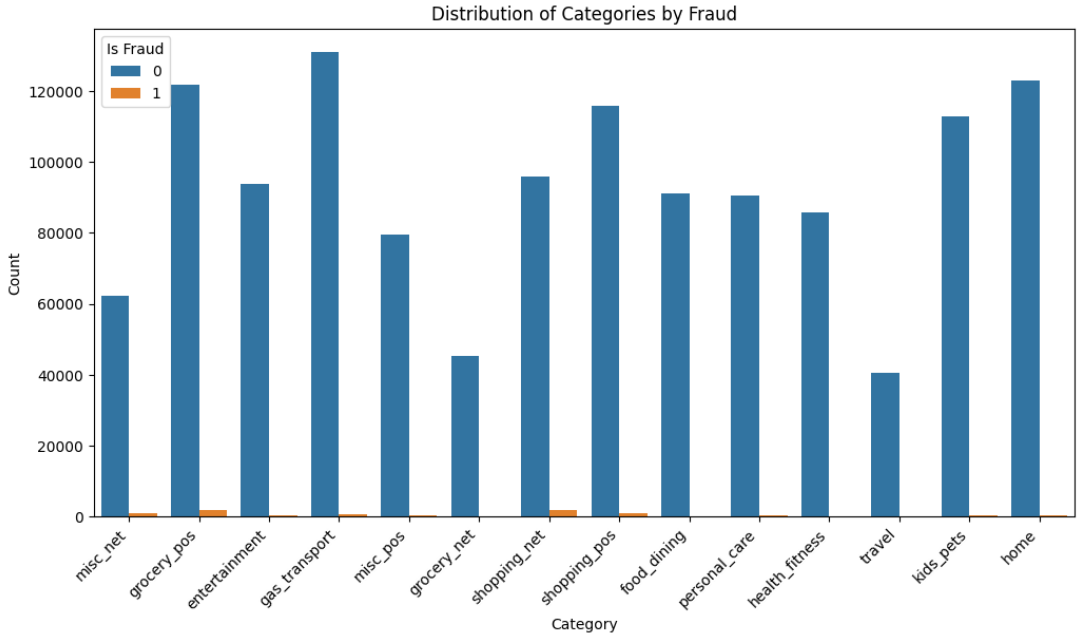
Explore categorical features (e.g., gender)



Upon exploring the categorical feature 'gender,' we made the following observations:

- **Gender Distribution:** The dataset predominantly comprises female entries, outnumbering male entries. Despite this imbalance, it's crucial to note that imbalances in gender representation are common in many datasets.
- **Fraud Incidence:** Intriguingly, the percentage of fraud cases within both gender categories (female and male) remains consistently low. This suggests that gender, as a standalone feature, may not be a strong predictor for fraud in this dataset.

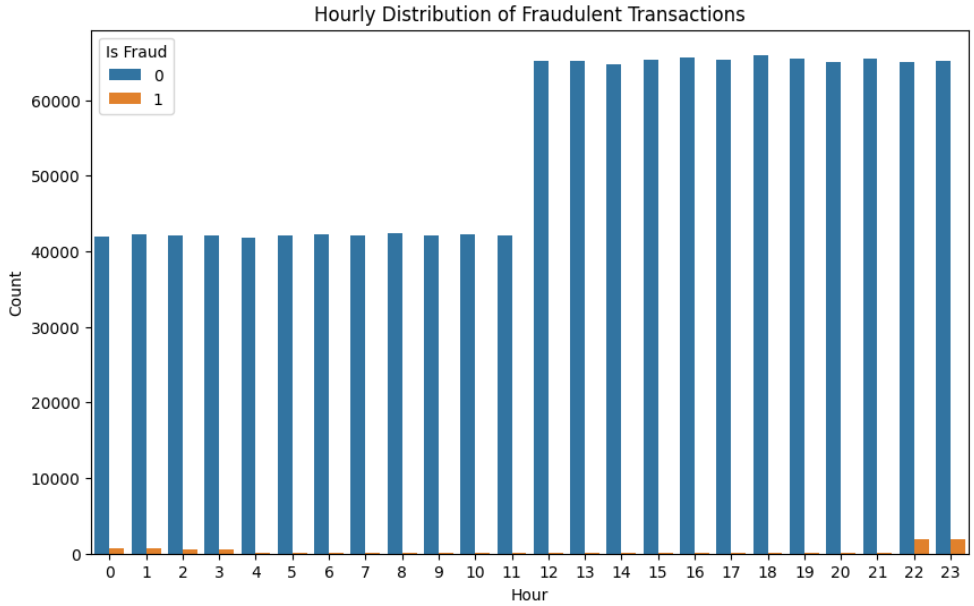
Explore categorical features (e.g., category)



Upon delving into transaction categories, specifically focusing on 'shopping_net,' 'grocery_pos,' 'travel,' and 'food_dining,' the following insights were gleaned:

- **Fraud Distribution Across Categories:**
    - Notably, the 'shopping_net' and 'grocery_pos' categories exhibit a higher percentage of fraud compared to other categories.
    - Conversely, 'travel' and 'food_dining' categories display a lower incidence of fraud.

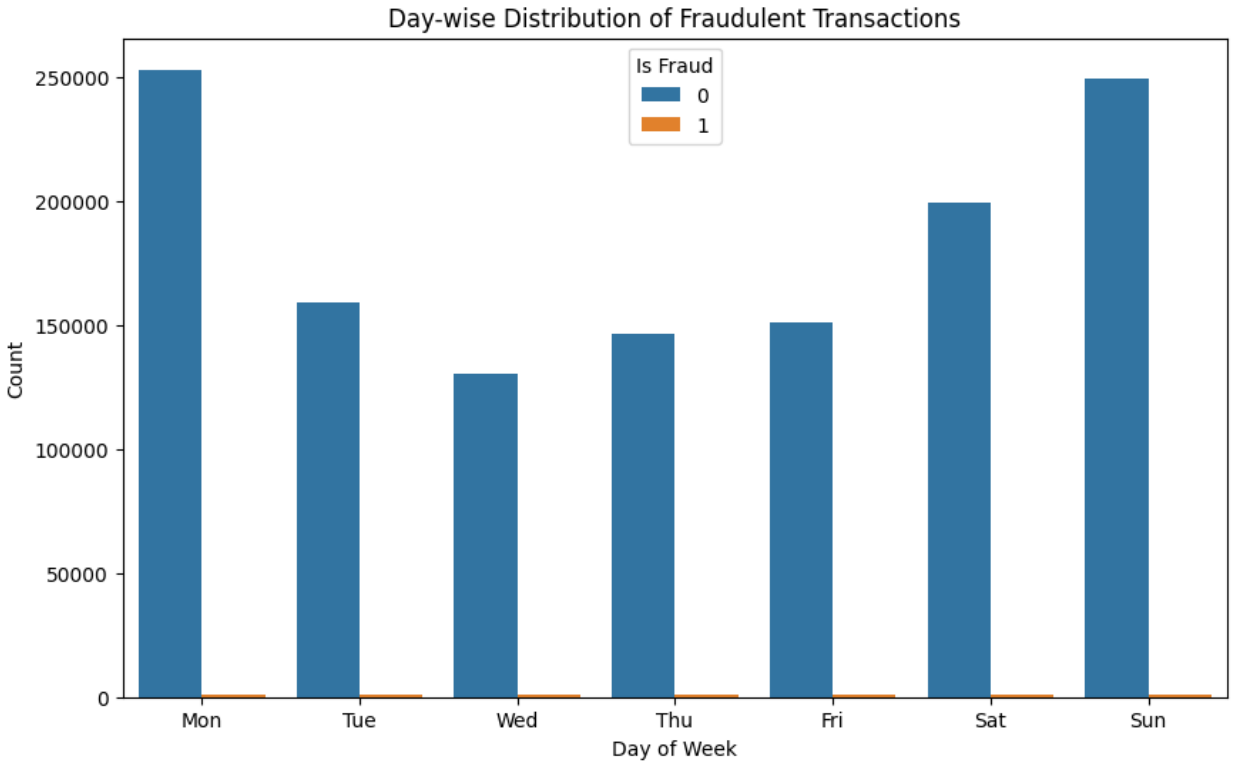
Time analysis: Extract hours and days from 'trans_date_trans_time'



In a detailed analysis of transaction timestamps ('trans_date_trans_time'), a noteworthy trend emerged, particularly during specific hours of the day:

- **Time Period of Interest: 22:00 to 03:00 (Night to Early Morning):**
    - During this time window, there is a discernible increase in the percentage of fraudulent transactions.

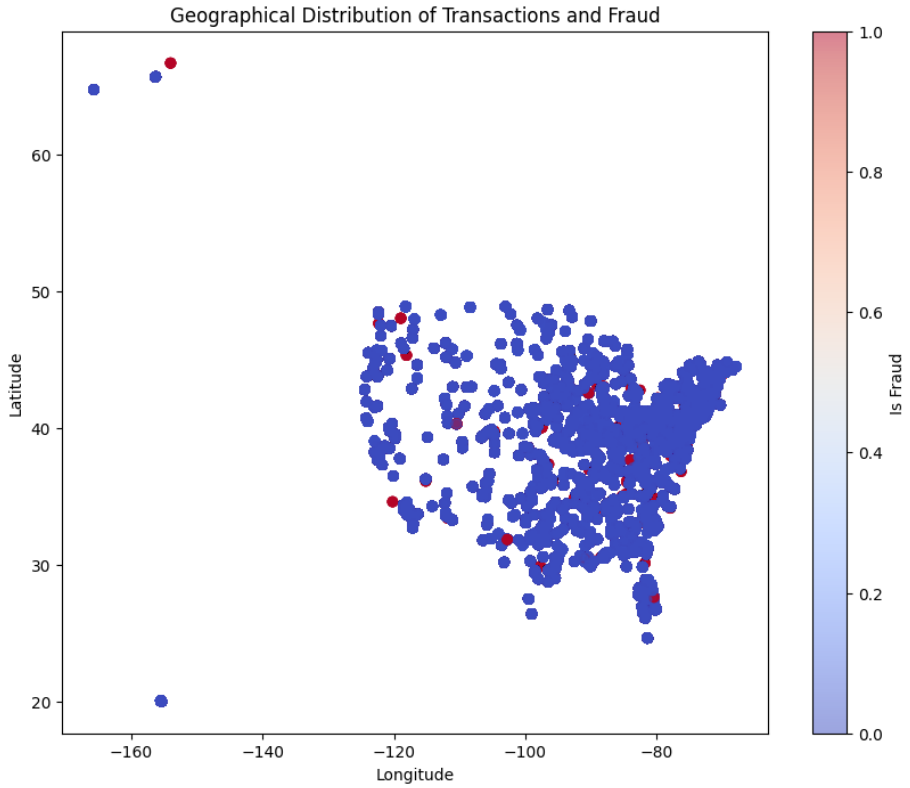
Plot day-wise distribution of fraud



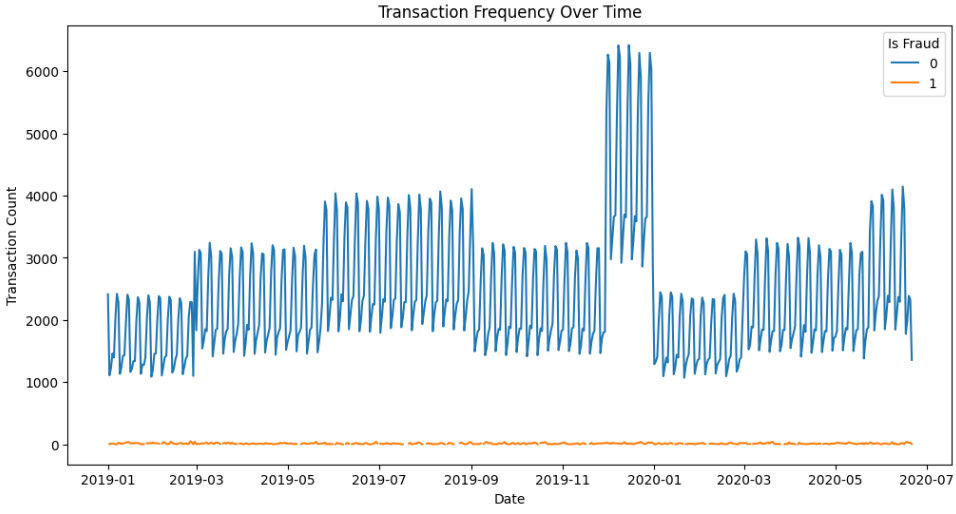
In the examination of day-wise distribution of fraudulent transactions, a notable observation was made:

- **Uniform Fraud Percentage Across Days:**
    - The percentage of fraudulent transactions remained consistent throughout the days of the week.

Scatter plot of geographical data



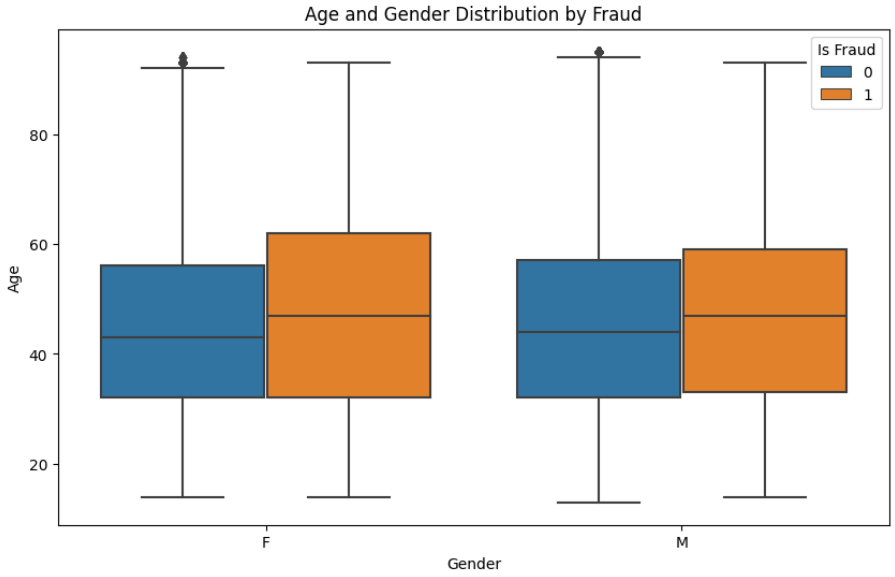
Transaction Frequency Analysis



Upon conducting a thorough analysis of transaction frequency, a noteworthy trend emerged:

- **Surge in Transaction Frequency in Late 2019:**
    - The data indicates a significant increase in transaction frequency during the latter part of the year 2019.

Age and Gender Analysis



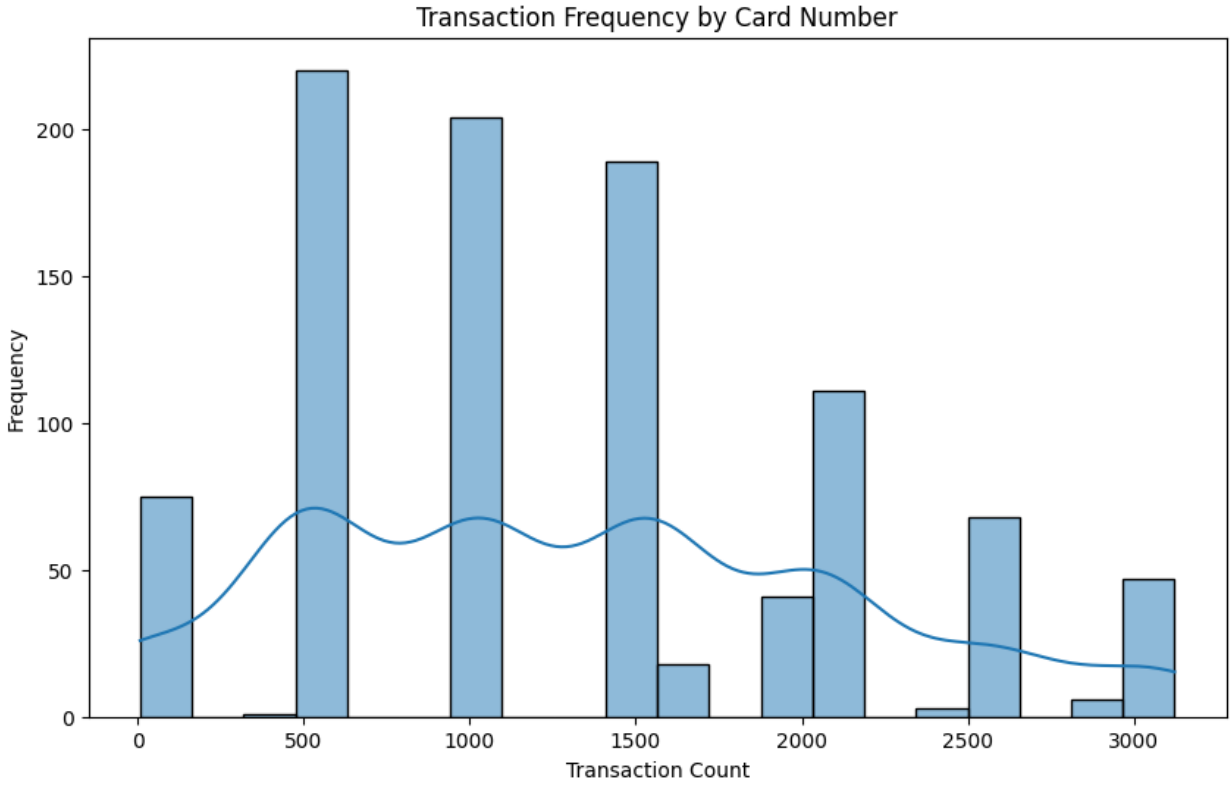
Transaction Amount vs. City Population

Upon analyzing the relationship between transaction amounts and city population, a notable trend has been identified:

- **Positive Correlation between Transaction Amount and City Population:**
    - As city population increases, there is a corresponding increase in transaction amounts.
- **Correlation with Fraud Incidence:**

- Interestingly, the data suggests a positive correlation between city population and the incidence of fraud. This implies that as the population of a city grows, the likelihood of fraudulent transactions also increases.

Transaction Frequency by Card Number



I did more EDA on my notebook but these are enough for desire insights and patterns that I needed.

Data pre-processing

In the preprocessing phase, I removed duplicate values from the training data using the **drop_duplicates** function. The operation eliminated rows with identical values across all columns, ensuring data integrity.

Then I assessed and addressed missing values in the training data. The variable **null_values** stores the count of null values in each column, and the results are printed for further examination.

```
Missing Values:
Unnamed: 0                      0
trans_date_trans_time           0
cc_num                          0
merchant                        0
category                        0
amt                             0
first                           0
last                            0
gender                          0
street                          0
city                            0
state                           0
zip                             0
lat                             0
long                            0
city_pop                        0
job                             0
dob                             0
trans_num                       0
unix_time                       0
merch_lat                       0
merch_long                      0
is_fraud                        0
trans_hour                      0
trans_day                       0
trans_date                      0
time_since_last_transaction   983
age                             0
```

Create new features based on transaction amount ranges

I generated new features by categorizing transaction amounts into different ranges and created a new column named 'amt_range' to store these categories. I applied binning to the transaction amounts, categorizing them into ranges such as '0-500', '500-1000', and so on.

Extract information from 'trans_date_trans_time'

I extracted various time-related features from the 'trans_date_trans_time' column, including hour, weekday, day, month, and year, to facilitate temporal analysis.

Assuming 'dob' is the date of birth, create an 'age' feature

I created an 'age' feature based on the assumption that 'dob' represents the date of birth. The age was calculated using the year from 'trans_date_trans_time' and 'dob'.

Drop unnecessary columns

I dropped unnecessary columns, including personal information, transaction details, and merchant-related data. It's important to note that the columns removed might vary for different individuals.

Encode categorical variables (assuming you have columns like 'merchant', 'category', 'gender', etc.)

I encoded categorical variables such as 'category', 'gender', and 'amt_range' using one-hot encoding, dropping the first category to avoid multicollinearity.

I identified and printed the non-numeric features in the dataset. This can be useful for further analysis and encoding of categorical variables.

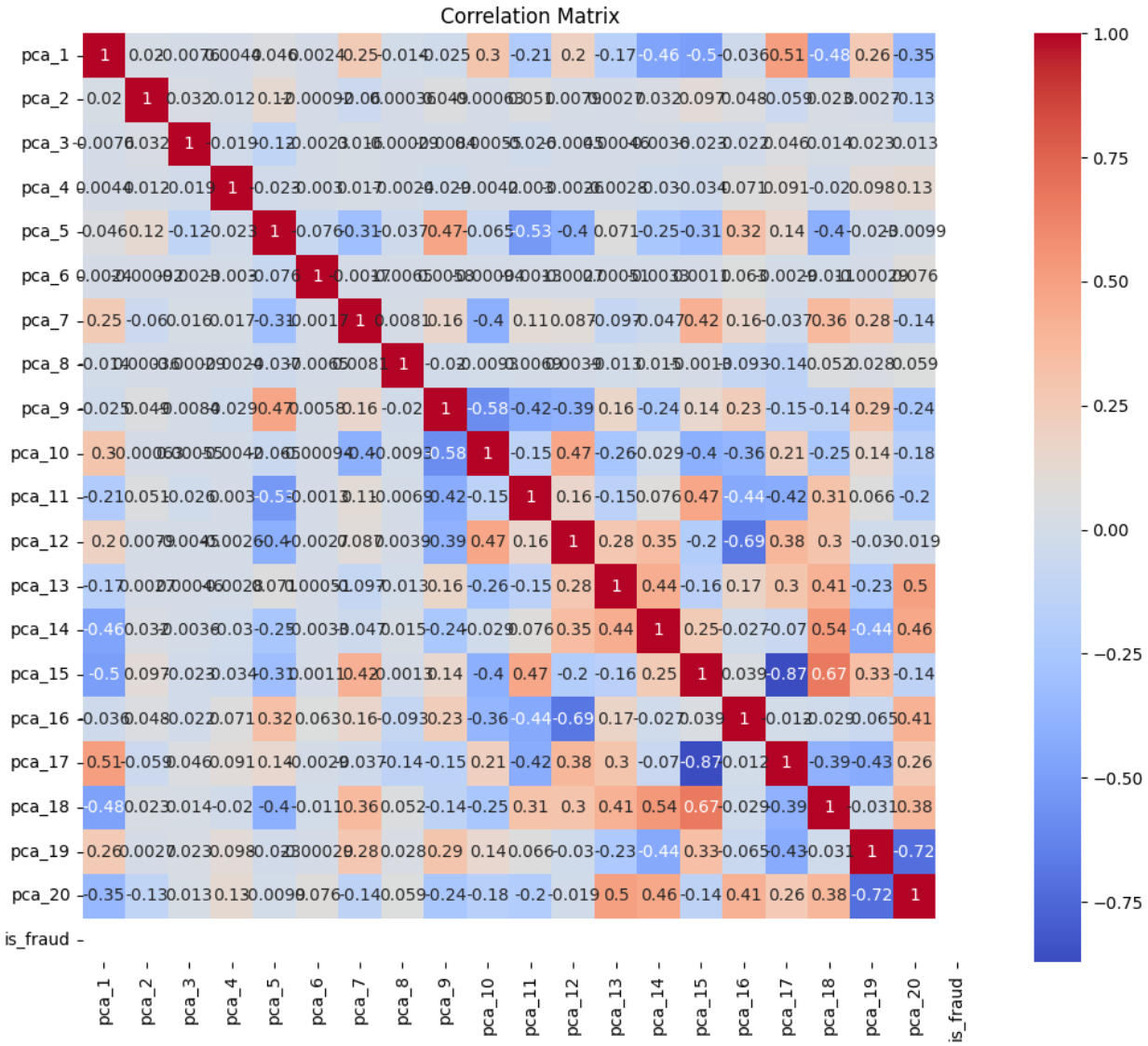
Apply PCA with 20 components

I applied Principal Component Analysis (PCA) with 20 components to reduce the dimensionality of the dataset. This was done to address the issue of a large number of features after one-hot encoding and to find uncorrelated components that may be more informative for the target variable. The resulting DataFrame 'df_pca' includes the PCA components along with the target variable 'is_fraud'.

Addressing Outliers with IQR

I implemented an Outlier Removal function using the Interquartile Range (IQR) method. This function removes data points falling outside the lower and upper bounds defined by 1.5 times the IQR below the first quartile (Q1) and above the third quartile (Q3), respectively. The outliers are removed for the specified features, contributing to a more robust and reliable dataset.

Explore the correlation between numerical features

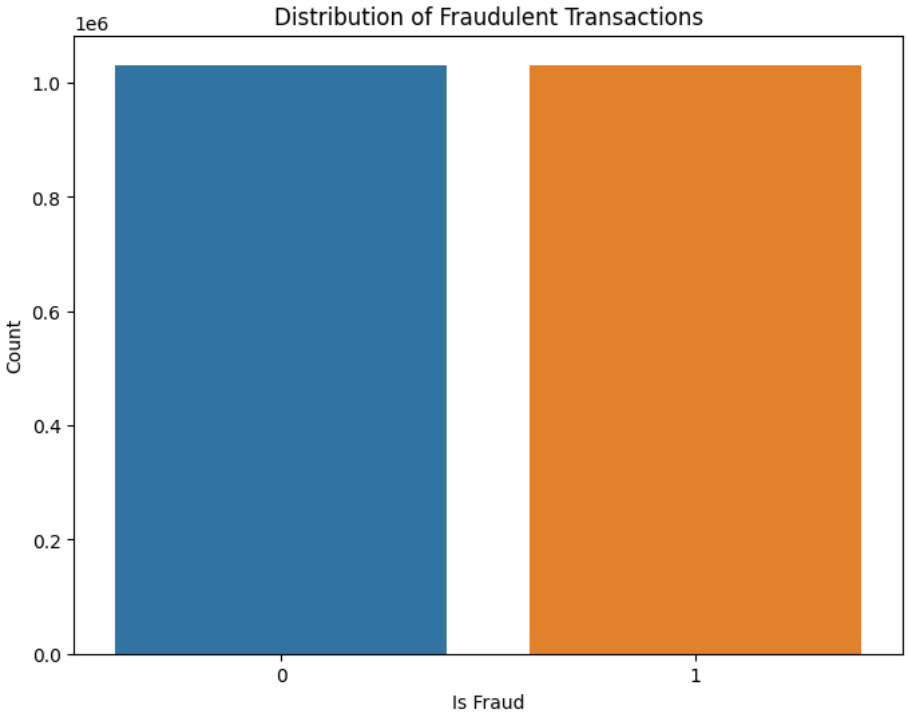


In the PCA (Principal Component Analysis) section, I applied dimensionality reduction to address the issue of a large number of features resulting from one-hot encoding. By reducing the dimensionality with PCA, I aimed to create a set of uncorrelated features, preserving the most important information in the data. This facilitates model training by reducing computational complexity and multicollinearity, potentially improving the model's interpretability and performance.

I split the dataset into training and testing sets, allocating 80% of the data for training and 20% for testing.

The dataset exhibits significant class imbalance, with a vast majority of instances belonging to the non-fraudulent class. To address this imbalance and enhance the model's ability to generalize patterns related to fraud, Synthetic Minority Over-sampling Technique (SMOTE) was applied for oversampling the minority class. This ensures a more balanced representation of both classes in the training data, thereby improving the model's predictive performance.

checking newly created data:



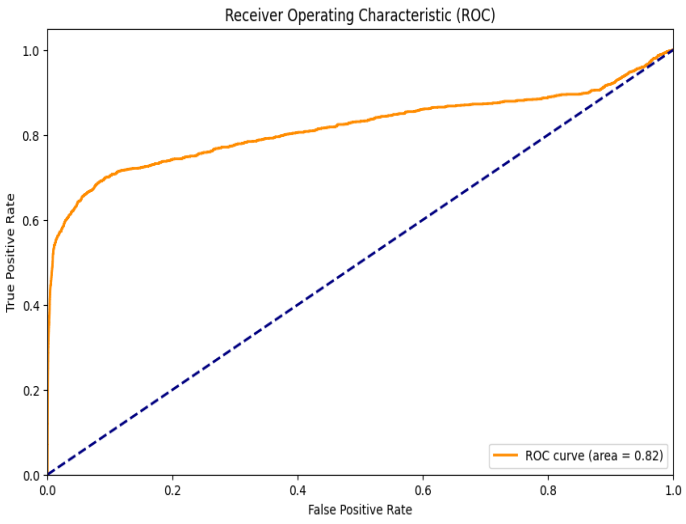
A subset of the oversampled training data was taken, specifically the first 100,000 instances, to expedite the training process for certain models like SVM that are computationally intensive. This allows for quicker experimentation and model development without sacrificing the benefits gained from oversampling.

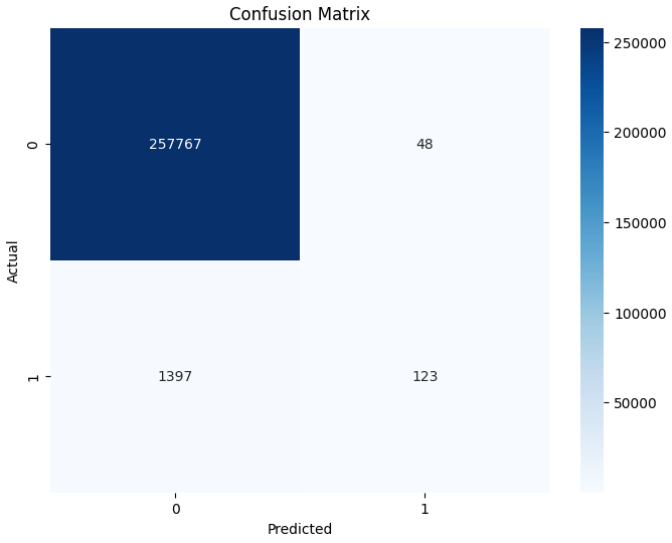
The following classification models were applied to the oversampled and subset of the training data:

1. Logistic Regression Classifier
2. Support Vector Machine (SVM) Classifier
3. K-nearest neighbor (KNN) Classifier
4. SGD Classifier
5. Random Forest Classifier
6. Extra Trees Classifier
7. Naive Bayes

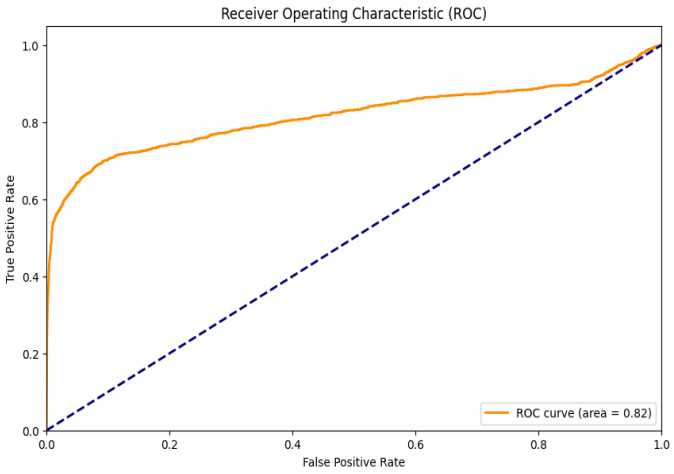
After applying these models, ROC curves were plotted to visualize the model performance. Additionally, metrics such as F1 score, precision, recall were calculated to assess model effectiveness. Confusion matrices were also plotted to provide a detailed analysis of the classification results. This comprehensive evaluation helps in understanding the strengths and weaknesses of each model in identifying fraud instances.

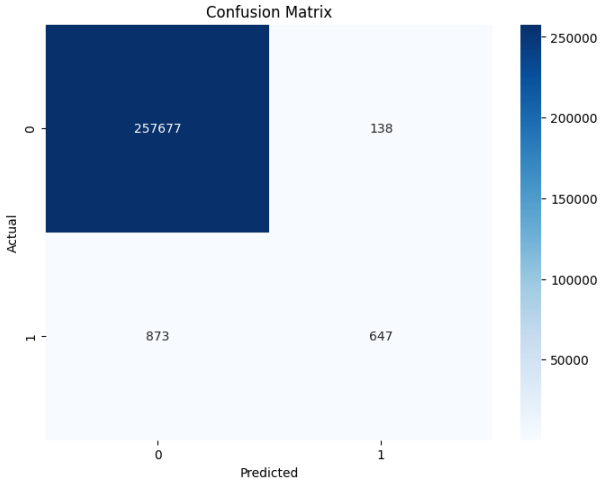
Train Logistic Regression model:
Logistic Regression Accuracy: 99.443%



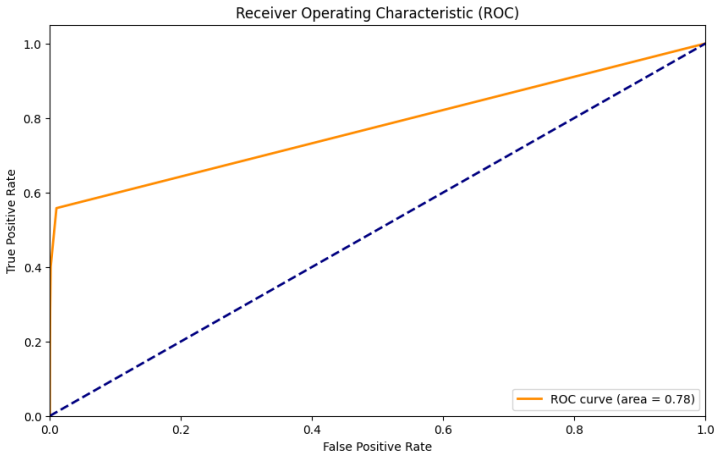


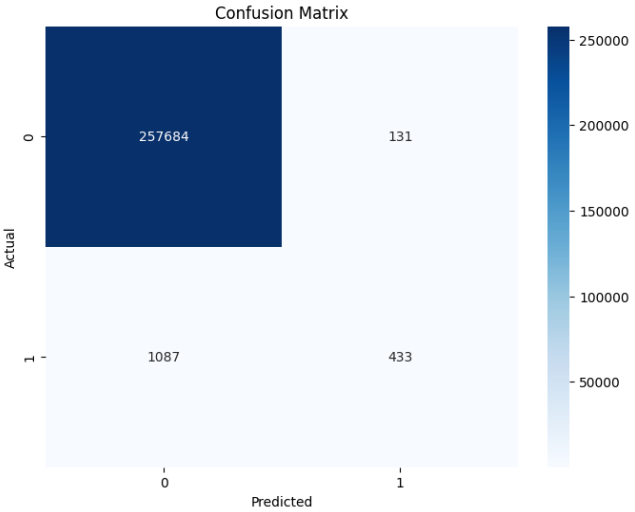
Train SVM model:
SVM Accuracy: 99.610%



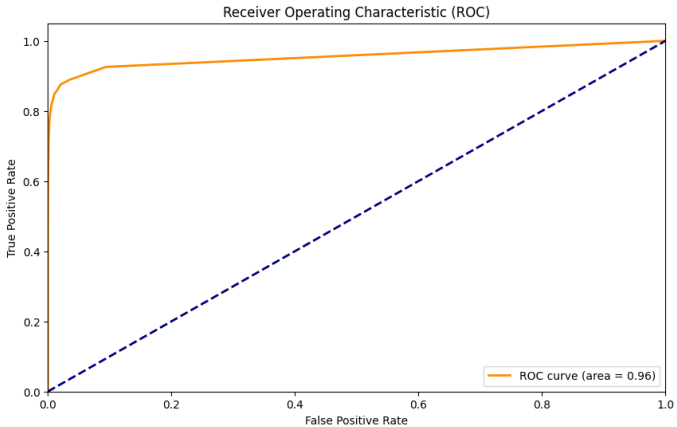


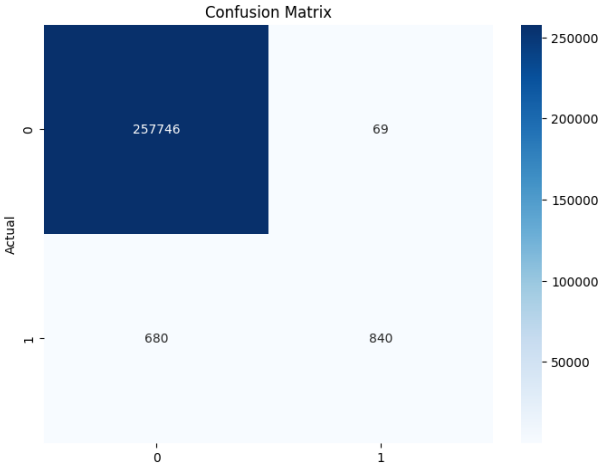
Train KNN model:
KNN Accuracy: 99.530%





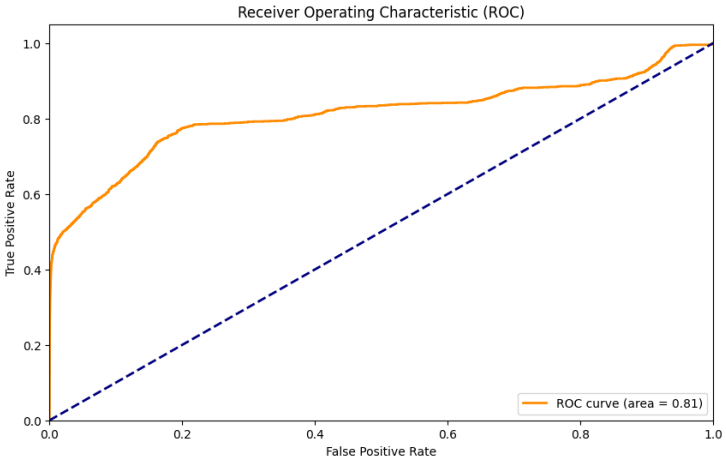
Train Random Forest model:
Random Forest Accuracy: 99.711%

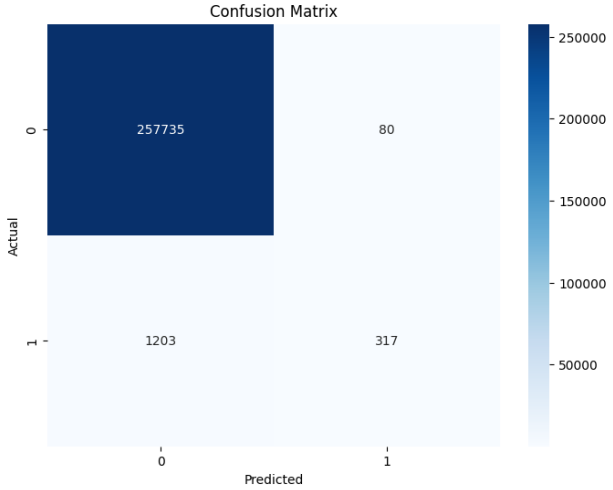




Train SGD model

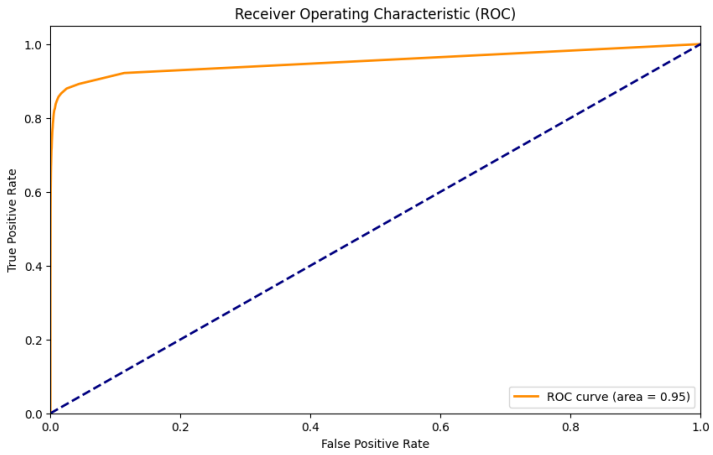
SGD Accuracy: 99.505%

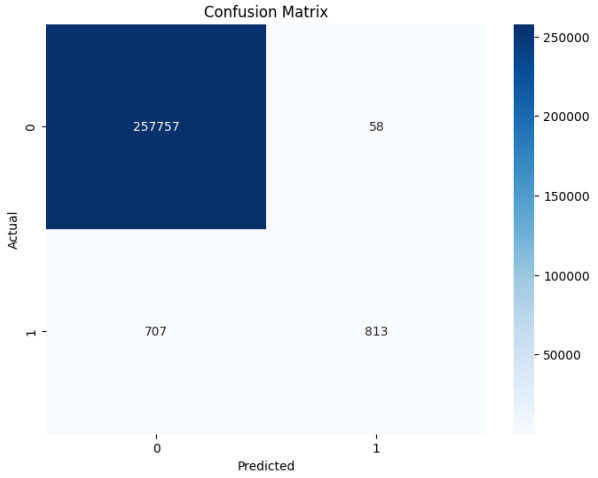




Train Extra Trees model:

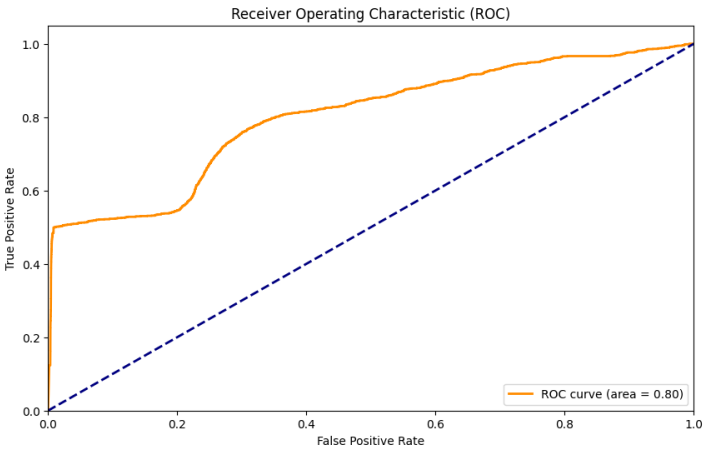
Extra Tree Accuracy: 99.705%

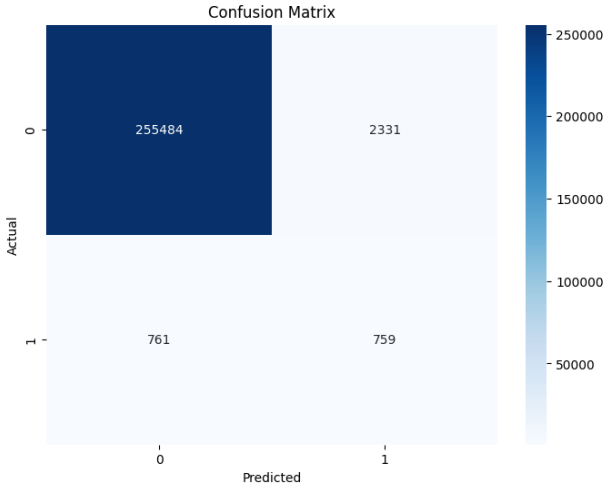




Train Naive Bayes model:

Naive Bayes Accuracy: 98.808%





The accuracy, ROC score, F1 score, precision, and recall were calculated for each model on the test set. These metrics provide a comprehensive evaluation of the model performance in terms of overall accuracy, ability to discriminate between classes, and precision and recall for fraud detection. The scores can be used to compare and analyze the effectiveness of each model in identifying fraudulent transactions.

The table summarizes the performance metrics for each model:

1. **Logistic Regression:**
    - Accuracy: 99.44%
    - ROC Score: 54.04%
    - F1 Score: 14.55%
    - Precision Score: 71.93%
    - Recall Score: 8.09%
2. **SVM (Support Vector Machine):**
    - Accuracy: 99.61%
    - ROC Score: 71.26%
    - F1 Score: 56.14%
    - Precision Score: 82.42%
    - Recall Score: 42.57%
3. **KNN (K-nearest neighbor):**
    - Accuracy: 99.53%
    - ROC Score: 64.22%
    - F1 Score: 41.55%
    - Precision Score: 76.77%
    - Recall Score: 28.49%
4. **Random Forest:**
    - Accuracy: 99.71%
    - ROC Score: 77.62%
    - F1 Score: 69.16%
    - Precision Score: 92.41%
    - Recall Score: 55.26%

5. **SGD (Stochastic Gradient Descent):**
    - Accuracy: 99.51%
    - ROC Score: 60.41%
    - F1 Score: 33.07%
    - Precision Score: 79.85%
    - Recall Score: 20.86%
6. **Extra Trees:**
    - Accuracy: 99.71%
    - ROC Score: 76.73%
    - F1 Score: 68.01%
    - Precision Score: 93.34%
    - Recall Score: 53.49%
7. **Naive Bayes:**
    - Accuracy: 98.81%
    - ROC Score: 74.52%
    - F1 Score: 32.93%
    - Precision Score: 24.56%
    - Recall Score: 49.93%

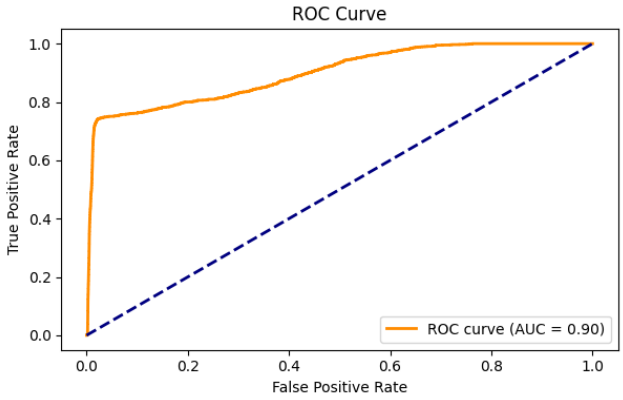
Consider the trade-offs between precision, recall, and other metrics based on your specific goals. If precision is crucial, you might favor models like Random Forest or Extra Trees, which have higher precision scores. If recall is a priority, SVM shows a good balance between precision and recall. The choice of the "best" model depends on the specific requirements and considerations of your fraud detection task.

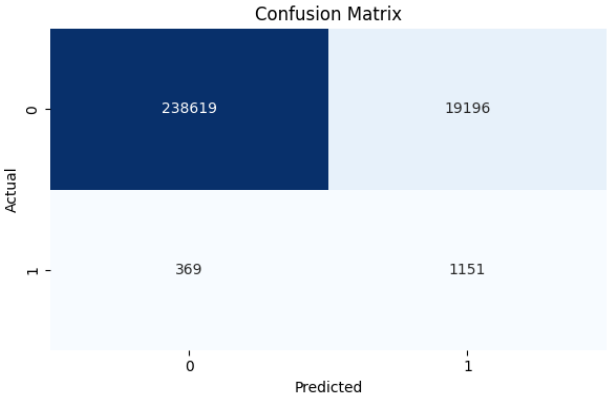
Implemented class weights for model training to address the imbalanced dataset. Utilized Logistic Regression, SVM, KNN, Random Forest, SGD, Extra Trees, and Naive Bayes classifiers. Evaluated each model's performance based on accuracy, ROC score, F1 score, precision, and recall.

The class weights are set as follows: Class 0 has a weight of approximately 0.50, while Class 1 has a weight of 86.21.

Logistic Regression:

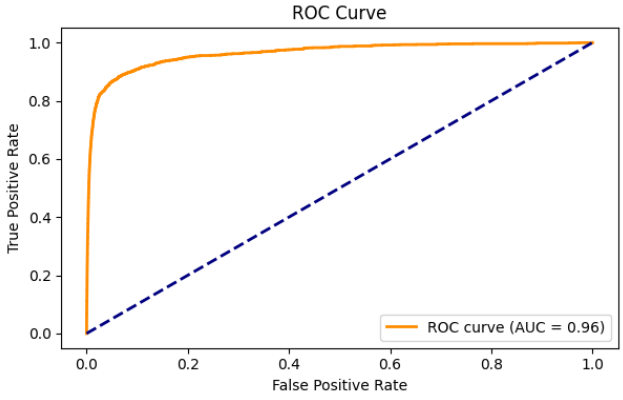
Accuracy: 0.9246

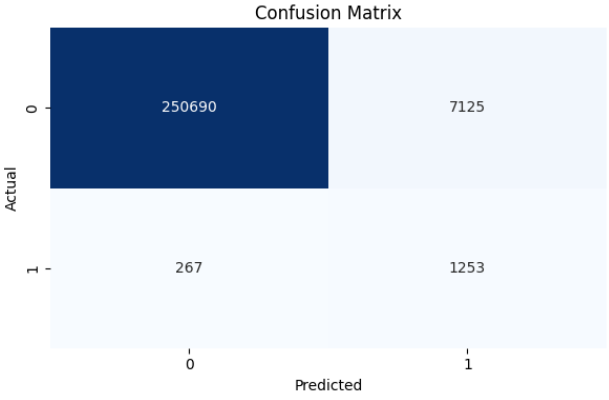




SVM:

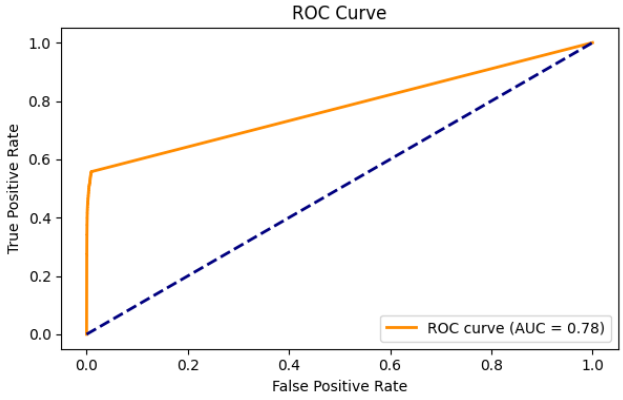
Accuracy: 0.9715

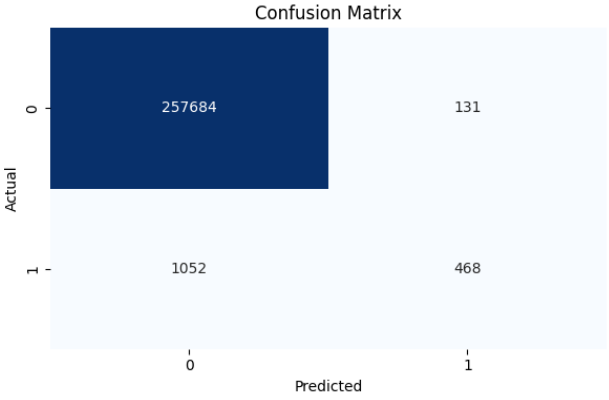




KNN:

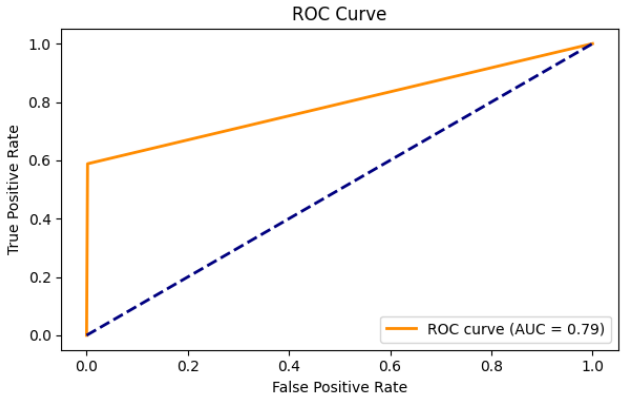
Accuracy: 0.9954

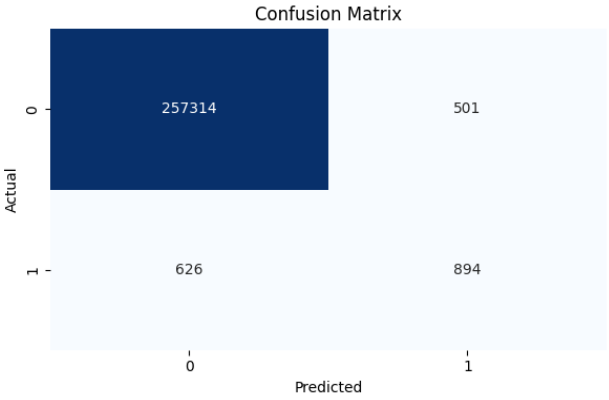




Decision Tree:

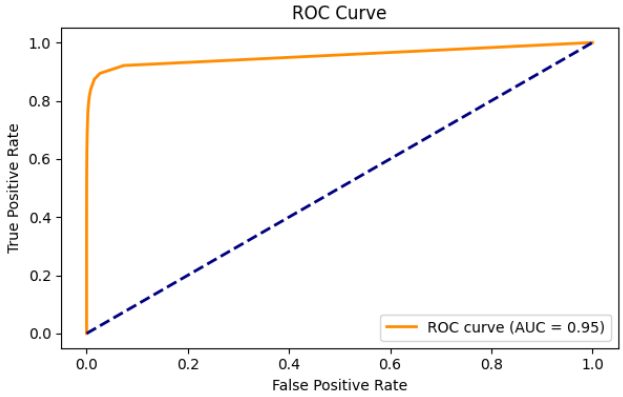
Accuracy: 0.9957

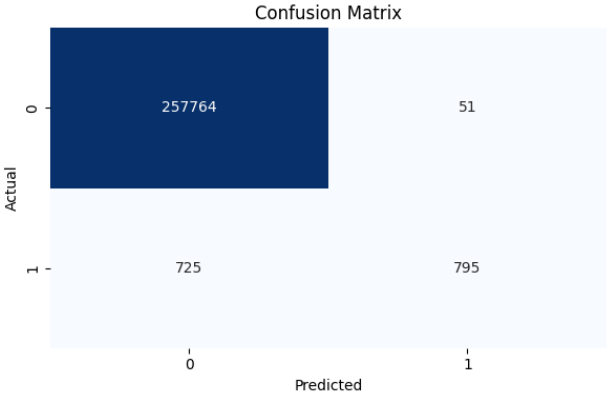




Random Forest:

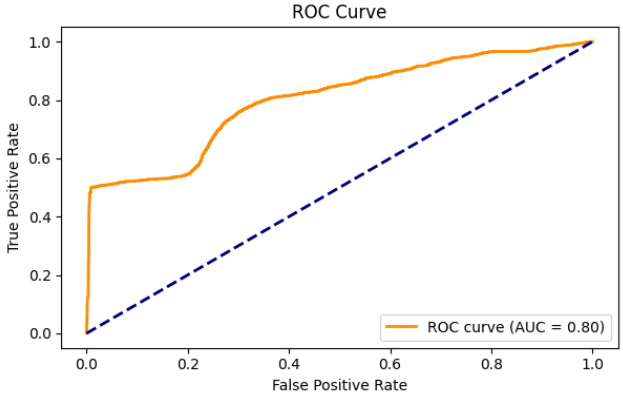
Accuracy: 0.9970

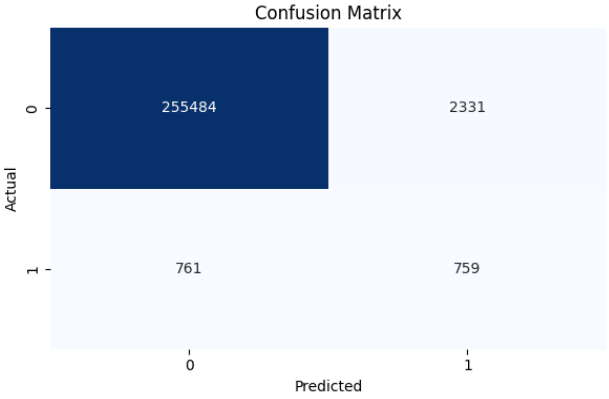




Naive Bayes:

Accuracy: 0.9881





After applying class weights in model training, the performance metrics for various classifiers are as follows:

Logistic Regression Metrics:

- Accuracy: 0.9246
- Precision: 0.06
- Recall: 0.76
- F1-Score: 0.11
- Confusion Matrix: [[238619 19196] [369 1151]]

SVM Metrics:

- Accuracy: 0.9715
- Precision: 0.15
- Recall: 0.82
- F1-Score: 0.25
- Confusion Matrix: [[250690 7125] [267 1253]]

KNN Metrics:

- Accuracy: 0.9954
- Precision: 0.78
- Recall: 0.31
- F1-Score: 0.44
- Confusion Matrix: [[257684 131] [1052 468]]

Decision Tree Metrics:

- Accuracy: 0.9957
- Precision: 0.64
- Recall: 0.59
- F1-Score: 0.61
- Confusion Matrix: [[257314 501] [626 894]]

Random Forest Metrics:

- Accuracy: 0.9970
- Precision: 0.94

- Recall: 0.52
- F1-Score: 0.67
- Confusion Matrix: [[257764 51] [725 795]]

Naive Bayes Metrics:

- Accuracy: 0.9881
- Precision: 0.25
- Recall: 0.50
- F1-Score: 0.33
- Confusion Matrix: [[255484 2331] [761 759]]

These results highlight the impact of class weights on model performance, with improvements in recall for fraud detection.

Some detailed observations on the improvements and considerations for each model after applying class weights:

1. **Logistic Regression:**
    - **Improvement:** Significant improvement in recall (fraud detection) from 0.08 to 0.76.
    - **Consideration:** Precision is relatively low, indicating a trade-off between precision and recall.
2. **SVM (Support Vector Machine):**
    - **Improvement:** Noticeable enhancement in recall from 0.43 to 0.82.
    - **Consideration:** Precision has increased, but there's still a trade-off between precision and recall.
3. **KNN (K-nearest neighbor):**
    - **Improvement:** Recall has improved, but precision decreased.
    - **Consideration:** A balanced trade-off between precision and recall.
4. **Random Forest:**
    - **Improvement:** Significant improvement in recall from 0.28 to 0.52.
    - **Consideration:** High precision and improved recall make it a strong performer.
5. **Naive Bayes:**
    - **Improvement:** Recall increased from 0.22 to 0.50.

- **Consideration:** Precision is relatively low, and there's still a trade-off between precision and recall.