PROBLEM STATEMENT

Credit cards are widely utilized for online purchases and payments, offering a convenient way to manage finances. However, they also carry inherent risks. Credit card fraud involves the unauthorized use of someone else's credit card or their card details to make purchases or withdraw cash. It is crucial for credit card companies to identify fraudulent transactions promptly, ensuring customers are not billed for charges they did not authorize.

ABOUT CREDIT CARD FRAUD DETECTION:

Credit card fraud detection encompasses the policies, tools, methodologies, and practices employed by credit card companies and financial institutions to combat identity fraud and prevent unauthorized transactions.

With the rapid increase in data volumes and the surge in payment card transactions, fraud detection has become predominantly digitized and automated. Today, many solutions rely on artificial intelligence (AI) and machine learning (ML) for data analysis, predictive modelling, decision-making, fraud alerts, and the subsequent remedial actions triggered by detected instances of credit card fraud.

Anomaly detection:

Anomaly detection involves analysing vast amounts of data from both internal and external sources to create a framework of "normal" behaviour for each user, helping to establish consistent activity patterns.

The data used to build user profiles typically includes:

- Purchase history and other historical data
- Location
- Device ID
- IP address
- Payment amount
- Transaction details

If a transaction deviates from the established norm, the anomaly detection system alerts the card issuer, and sometimes the user. Based on the transaction details and the associated risk score, the system may either flag the transaction for further review or place a temporary hold until the user confirms their activity.

What qualifies as an anomaly?

- A sharp increase in spending
- Purchasing high-value items
- Rapid succession of transactions
- Multiple purchases from the same vendor
- Transactions originating from an unfamiliar location or overseas
- Transactions made at unusual hours

When anomaly detection systems incorporate machine learning (ML), they can become self-adapting. This means the system continually collects and processes new data to refine its model, improving the accuracy of detecting what constitutes typical behaviour for each user over time.

INTRODUCTION:

The dataset contains transactions made by credit cards in September 2013 by European cardholders. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

In this Project, we have to build a classification model to predict whether a transaction is fraudulent or not. We will use various predictive models to see how accurate they are in detecting whether a transaction is a normal payment or a fraud. Let's start!

PROJECT OUTLINE:

- **Exploratory Data Analysis:** Analysing and understanding the data to identify patterns, relationships, and trends in the data by using Descriptive Statistics and Visualizations.
- **Data Cleaning:** Checking for the data quality, handling the missing values and outliers in the data.
- **Dealing with Imbalanced data:** This data set is highly imbalanced. The data should be balanced using the appropriate Resampling Techniques (NearMiss Undersampling, SMOTETomek) before moving onto model building.
- **Feature Engineering:** Transforming the existing features for better performance of the ML Models.
- **Model Training:** Splitting the data into train & test sets and use the train set to estimate

the best model parameters.

- **Model Validation:** Evaluating the performance of the models on data that was not used during the training process. The goal is to estimate the model's ability to generalize to new, unseen data and to identify any issues with the model, such as overfitting.
- **Model Selection:** Choosing the most appropriate model that can be used for this project.
- **Model Deployment:** Model deployment is the process of making a trained machine learning model available for use in a production environment.

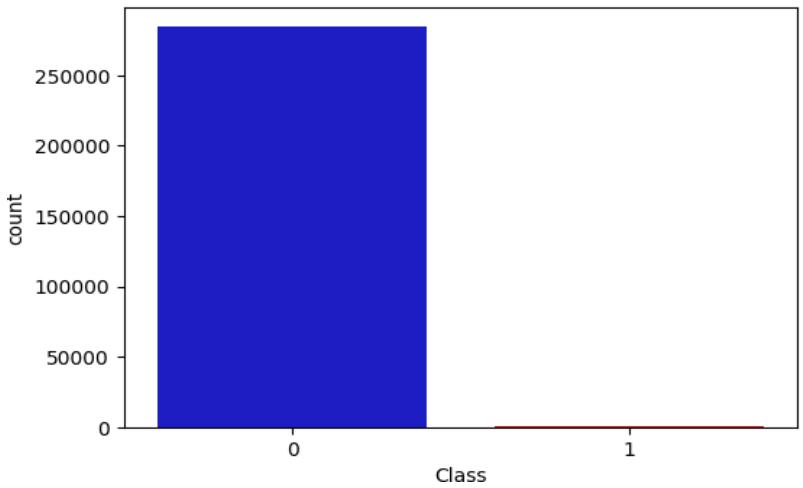
PROJECT WORK:

Exploratory Data Analysis:

The dataset contains transactions made by credit cards in September 2013 by European cardholders. This dataset presents transactions that occurred in two days, where we have 492 frauds out of 284,807 transactions. The dataset is highly unbalanced, the positive class (frauds) account for 0.172% of all transactions.

Below is the distribution of Feature 'Class' which is the response variable, and it takes value 1 in case of fraud and 0 otherwise.

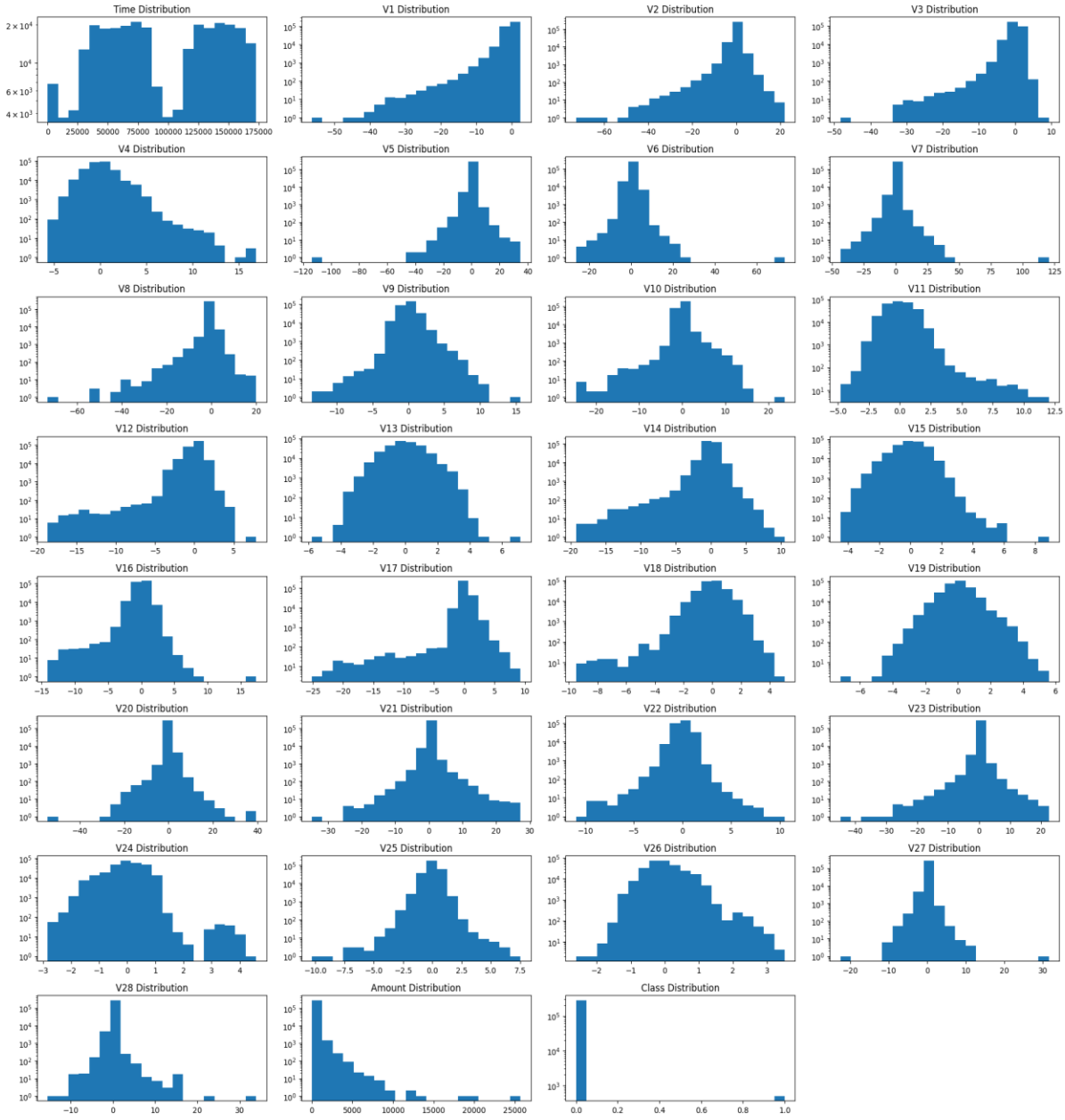
Count Plot for the Class Distribution



Our Credit card dataset contains only numerical input variables which are the result of a PCA transformation. Unfortunately, due to confidentiality issues the original features and more background information about the data is not provided. Features V1, V2, … V28 are the principal components obtained with PCA, the only features which have not been transformed with PCA are 'Time' and 'Amount'. Feature 'Time' contains the seconds elapsed between each transaction and the first transaction in the dataset. The feature 'Amount' is the transaction Amount, this feature can be used for example-dependant cost-sensitive learning.

Dataset has been examined for the missing values, Notably, there are no missing values present in the dataset.

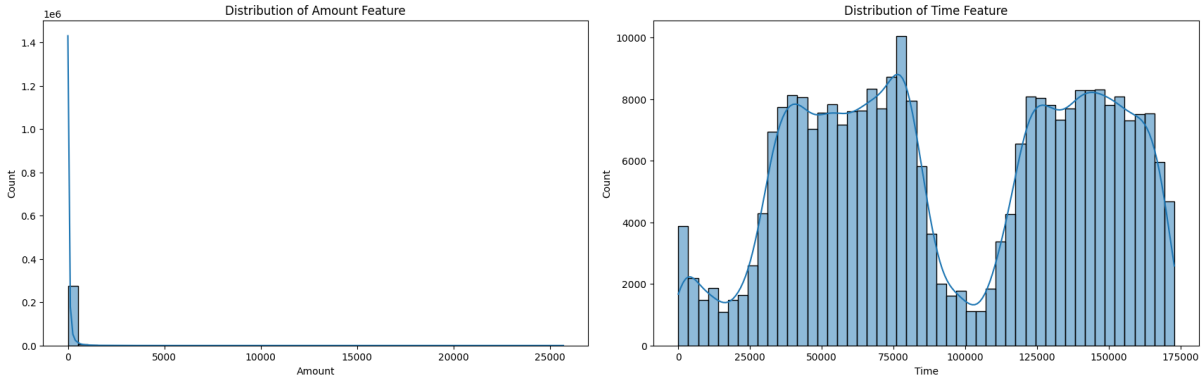
Below are the distributions for each feature -



The columns labeled 'V1' to 'V28' have undergone transformation using PCA techniques,

while the 'Class' column serves as the target variable. As a result, our primary focus will be on analysing the 'Time' and 'Amount' columns.

Distributions plots for Transaction Amount and Time



Based on the distplots illustrating the distribution of Transaction Amount and Time, notable skewness is observed in both columns. The Transaction Amount feature exhibits a right-skewed distribution, indicating a higher frequency of smaller transactions with a tail extending towards larger values. As for the Time distribution, it illustrates transaction timings over a two-day period. Notably, transactions appear to be least frequent during nighttime and peak during daytime hours.

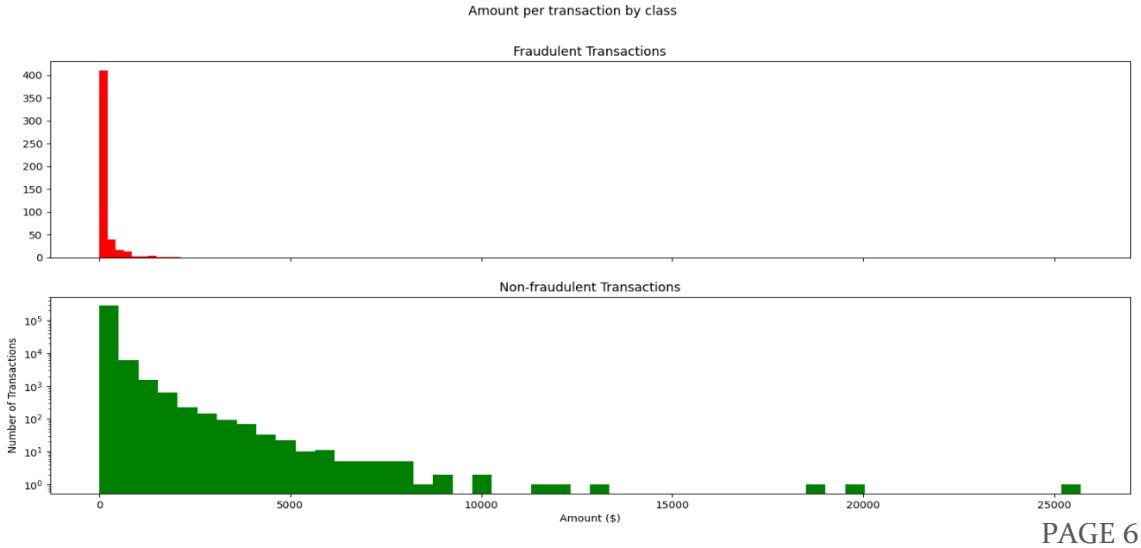
After segregating the dataset based on the class label into fraudulent and non-fraudulent dataframes, the following is the description:

Fraudulent Transactions

```
count     492.000000
mean      122.211321
std       256.683288
min         0.000000
25%         1.000000
50%         9.250000
75%       105.890000
max      2125.870000
Name: Amount, dtype: float64
```

Non-fraudulent Transactions

```
count    284315.000000
mean         88.291022
std         250.105092
min           0.000000
25%           5.650000
50%          22.000000
75%          77.050000
max       25691.160000
Name: Amount, dtype: float64
```



Upon examining the above description and distribution of fraudulent and non-fraudulent amounts, it becomes evident that the range of fraudulent amounts (0 to 2125) is narrower compared to non-fraudulent amounts (0 to 25691). Additionally, fraudulent amounts exhibit higher mean and standard deviation values.

Overall after analyzing our dataset we can conclude that the dataset exhibits significant class imbalance, with the majority of transactions being non-fraudulent (99.82%). Using this dataset as the foundation for our predictive models and analysis may lead to substantial errors and overfitting. This is because the algorithms may incorrectly assume that most transactions are not fraudulent, compromising their ability to detect genuine fraud patterns. Instead, we aim for our model to discern distinctive patterns indicative of fraudulent activity rather than making assumptions based on class distribution.

Data Preprocessing:

As the columns labeled V1 to V28 contain sensitive information and have already undergone both scaling and PCA transformation, there is limited opportunity for further feature transformation or analysis with respect to these features.

Time and Amount, like other columns, require scaling for consistency in data preprocessing. We have opted to scale these features using RobustScaler, a technique commonly employed in handling skewed or outlier-prone data distributions. This ensures that Time and Amount are appropriately transformed along with the other columns, contributing to a more uniform and standardized dataset.

Along with this we will further use resampling techniques to mitigate the impact of the class imbalance.

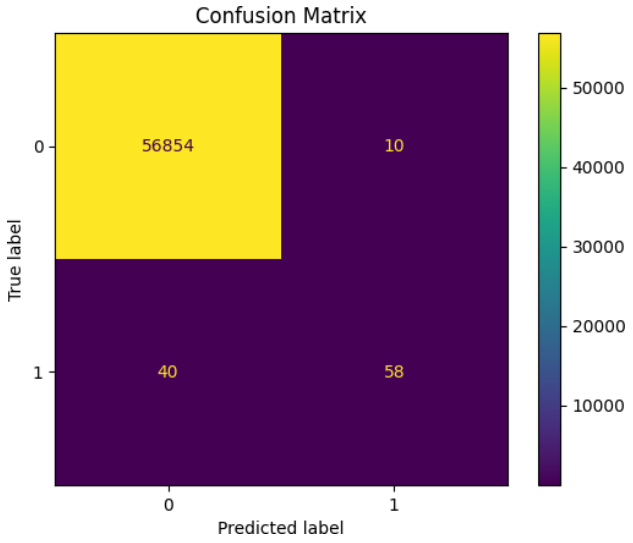
Splitting the data into Training and Testing sets:

Before initiating the Resampling techniques, it's essential to isolate the original dataframe. This step is crucial because, for testing purposes, our objective is to evaluate the model's performance on the original testing set, not on a testing set generated by undersampling or oversampling techniques. Although we split the data when applying UnderSampling or OverSampling techniques, our aim is to train the model using the undersampled or oversampled dataframes to enable pattern detection. However, the model should ultimately be tested on the original testing set to ensure accurate evaluation.

Model Training on Imbalanced Dataset:

Following the division of the dataset into training and testing subsets, we proceeded to evaluate the performance of Logistic Regression and RandomForest Classifier models on this imbalanced dataset. This assessment aimed to gauge the effectiveness of these models in handling the inherent class imbalance present in the data, providing insights into their suitability for credit card fraud detection tasks.

Logistic Regression model results:



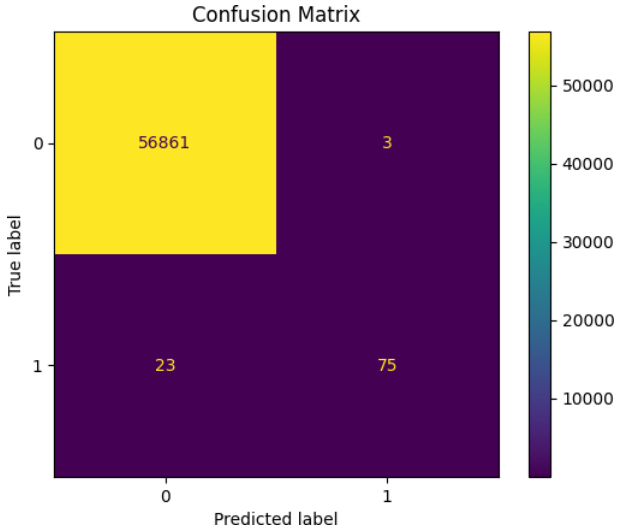
Based on the Confusion Matrix analysis, it is evident that the LR model performs well in predicting non-fraudulent transactions (label 0), as indicated by the high number of true negatives (TN). However, concerning fraudulent transactions (total 98), the model's performance is less satisfactory. Specifically, out of the 98 actual fraudulent transactions, the model correctly identifies 58 (true positives, TP). However, it misclassifies 40 fraudulent transactions as non-fraudulent (false negatives, FN), which is a significant concern.

```
Testing Accuracy:  0.9991222218320986

              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
           1       0.85      0.59      0.70        98

    accuracy                           1.00     56962
   macro avg       0.93      0.80      0.85     56962
weighted avg       1.00      1.00      1.00     56962
```

While the Logistic Regression model achieves a high accuracy score of 0.9991, caution is warranted due to the highly imbalanced nature of the data. The lower precision (0.85), recall (0.59), and F1-score (0.70) for the (label 1) fraudulent class indicate a struggle in correctly identifying fraudulent transactions. Although weighted and macro average scores are higher due to the dominance of the majority class, this doesn't necessarily imply good model fit. Therefore, further adjustments or alternative approaches may be necessary to improve performance, especially in capturing the characteristics of fraudulent transactions.

Random Forest Classifier results:

It is one of the ensemble techniques, which inherently handle class imbalance better than individual models.



As compared to the Logistic Regression model, RF classifier has shown significant improvement in predicting fraudulent transactions. We can see model has correctly classified 75 transactions(TP) out of total 98 fraudulents where LR's TP is 58. FN (23) & FP (3) has decreased in RFC compared to LR (40) & (10) respectively.

```
Testing Accuracy:  0.9995435553526912

              precision    recall  f1-score   support

           0       1.00      1.00      1.00     56864
           1       0.96      0.77      0.85        98

    accuracy                           1.00     56962
   macro avg       0.98      0.88      0.93     56962
weighted avg       1.00      1.00      1.00     56962
```

Logistic Regression model Vs Random Forest Classifier-

- Testing Accuracy: The testing accuracy of 0.9995 is slightly higher than the logistic reg accuracy of 0.9991.
- Precision, Recall, and F1-score: For the fraudulent class (label 1), precision has increased from 0.85 to 0.96, recall has increased from 0.59 to 0.77, and the F1-score has increased from 0.70 to 0.85.
- Macro Average: The macro-average precision, recall, and F1-score are 0.98, 0.88, and 0.93 respectively, compared to 0.93, 0.80, and 0.85 in the previous summary.

Overall, the new evaluation metrics show a slight improvement in accuracy and precision for the fraudulent class, while recall has notably increased. The macro-average metrics also show an improvement, indicating a better balance between precision and recall across both classes. However, weighted-average metrics remain unchanged, reflecting there's a still dominance of the non-fraudulent class in the dataset.

Handling Class Imbalance:

In our credit card dataset, the non-fraudulent class significantly outweighs the fraudulent class, resulting in imbalanced data leading to biased models that favor the majority (non-fraudulent) class. Handling this class imbalance is essential for ensuring the effectiveness of our models. Several techniques can help mitigate this issue:

1. Resampling Techniques:
    - Undersampling: Reducing the size of the majority class to match the minority class.
    - Oversampling: Increasing the size of the minority class by duplicating samples or generating synthetic samples.
    - Advanced Sampling Techniques: Using advanced techniques like SMOTE (Synthetic Minority Over-sampling Technique) or ADASYN (Adaptive Synthetic Sampling) for oversampling the minority class with more sophistication.
2. Model Tuning: Fine-tuning model hyperparameters to optimize performance.
3. Ensemble Methods: Harnessing the power of ensemble techniques to combine multiple models for improved accuracy.

Performing Undersampling using NearMiss

NearMiss is an undersampling technique commonly used to address class imbalance problems in machine learning classification tasks. It aims to balance the class distribution by reducing the number of samples in the majority class, making it more comparable to the minority class.

Types of NearMiss: There are several variations of NearMiss, including NearMiss-1, NearMiss-2, and NearMiss-3. Each variant employs a different criterion for selecting samples from the majority class based on their proximity to minority class samples.

- NearMiss-1: NearMiss-1 selects samples from the majority class for which the average distance to the k nearest neighbors in the minority class is the smallest.
- NearMiss-2: NearMiss-2 selects samples from the majority class by focusing on the farthest samples from the decision boundary between the two classes. It retains samples that are closest to the minority class but farthest from the majority class.
- NearMiss-3: NearMiss-3 is similar to NearMiss-2 but considers a different criterion for selecting samples. It retains samples from the majority class that are closest to the centroids of the minority class.

4. To mitigate the class imbalance in our dataset, we utilized the NearMiss-1 undersampling technique on our training dataset. This method equalizes the class distribution by decreasing the instances of Non-fraudulent Transactions to align with the count of Fraudulent Transactions.

5. Before applying the NearMiss-1 technique, the class distribution was as follows: Counter ({0: 227451, 1: 394}). After applying the undersampling technique, the class distribution became Counter ({0: 394, 1: 394}). This adjustment ensures a more balanced representation of both classes, which is crucial for training effective machine learning models to detect fraudulent transactions.

Correlation matrices:

Correlation matrices play a crucial role in understanding our data, especially in identifying features that significantly influence whether a transaction is fraudulent. It's essential to use the correct dataframe (such as a subsample) to analyse which features have a strong positive or negative correlation with fraud transactions.

Key Points:

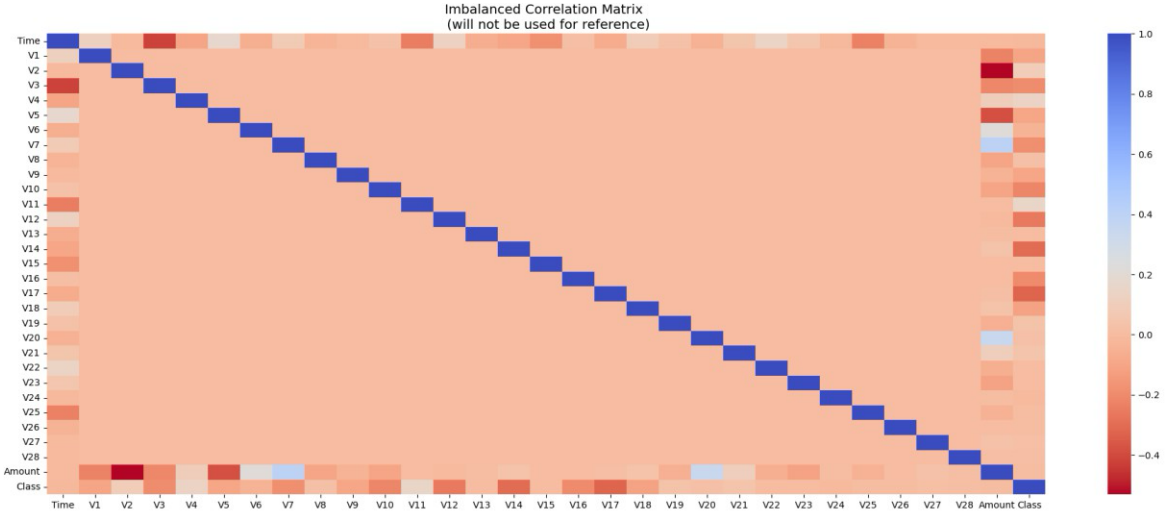
1. Negative Correlations: Features like V17, V14, V12, and V10 exhibit negative correlations. Lower values in these features tend to be associated with a higher likelihood of a transaction being fraudulent.
2. Positive Correlations: Features such as V2, V4, V11, and V19 show positive correlations with fraud transactions. Higher values in these features are indicative of a higher probability of the transaction being fraudulent.

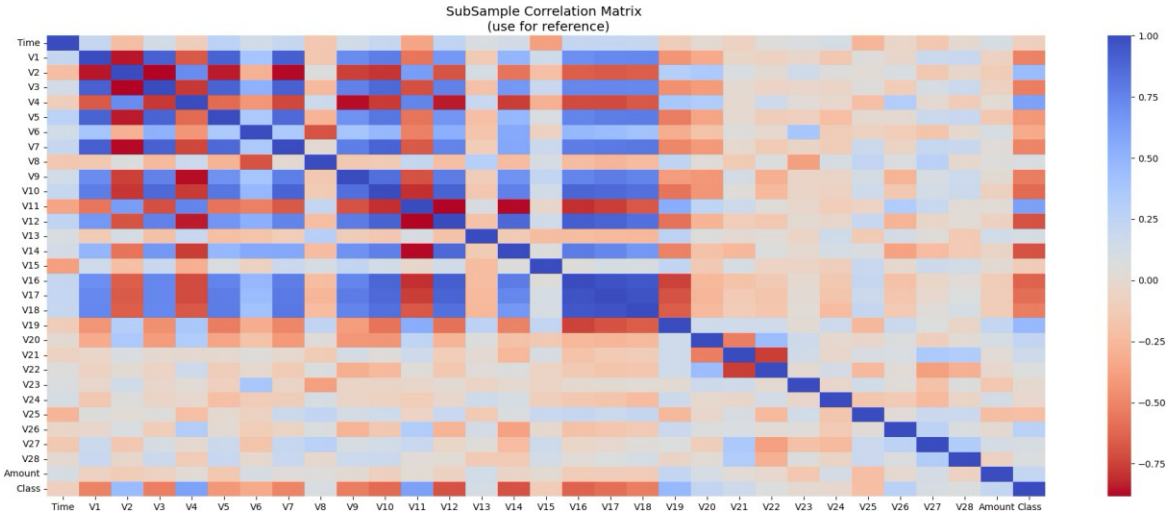
Correlation matrices:

Correlation matrices play a crucial role in understanding our data, especially in identifying features that significantly influence whether a transaction is fraudulent. It's essential to use the correct dataframe (such as a subsample) to analyse which features have a strong positive or negative correlation with fraud transactions.

Key Points:

3. Negative Correlations: Features like V17, V14, V12, and V10 exhibit negative correlations. Lower values in these features tend to be associated with a higher likelihood of a transaction being fraudulent.
4. Positive Correlations: Features such as V2, V4, V11, and V19 show positive correlations with fraud transactions. Higher values in these features are indicative of a higher probability of the transaction being fraudulent.

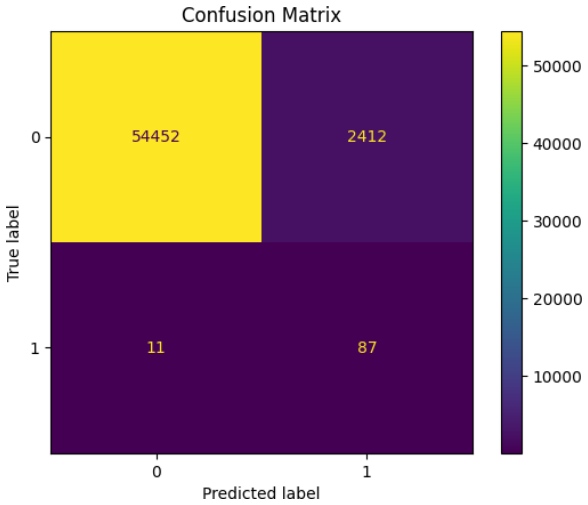

Imbalanced Correlation Matrix
(will not be used for reference)
Time
V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
V11
V12
V13
V14
V15
V16
V17
V18
V19
V20
V21
V22
V23
V24
V25
V26
V27
V28
Amount
Class
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4


SubSample Correlation Matrix
(use for reference)
Time
V1
V2
V3
V4
V5
V6
V7
V8
V9
V10
V11
V12
V13
V14
V15
V16
V17
V18
V19
V20
V21
V22
V23
V24
V25
V26
V27
V28
Amount
Class
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75

Results of Models trained on Undersampled data:

Following undersampling of the dataset, various classification models were trained using the balanced data. These models include Logistic Regression, K-Nearest Neighbors (KNN), Random Forest Classifier, as well as boosting models like AdaBoost and XGBoost. The results of these models are presented below.

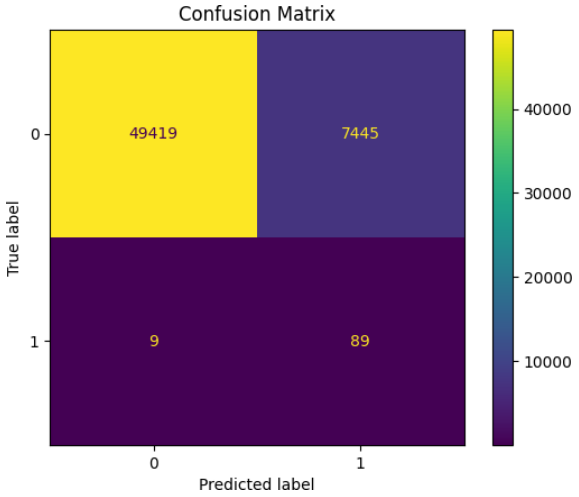
Logistic Regression Model-



Testing Accuracy: 0.9574628699834977

	precision	recall	f1-score	support
0	1.00	0.96	0.98	56864
1	0.03	0.89	0.07	98
accuracy			0.96	56962
macro avg	0.52	0.92	0.52	56962
weighted avg	1.00	0.96	0.98	56962

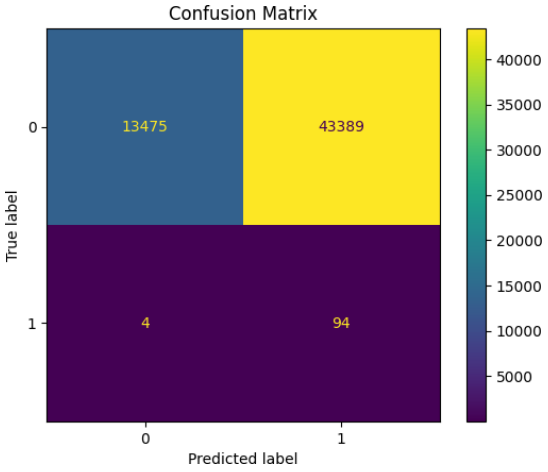
K-Nearest Neighbors (KNN) model-



Testing Accuracy: 0.8691408307292581

	precision	recall	f1-score	support
0	1.00	0.87	0.93	56864
1	0.01	0.91	0.02	98
accuracy			0.87	56962
macro avg	0.51	0.89	0.48	56962
weighted avg	1.00	0.87	0.93	56962

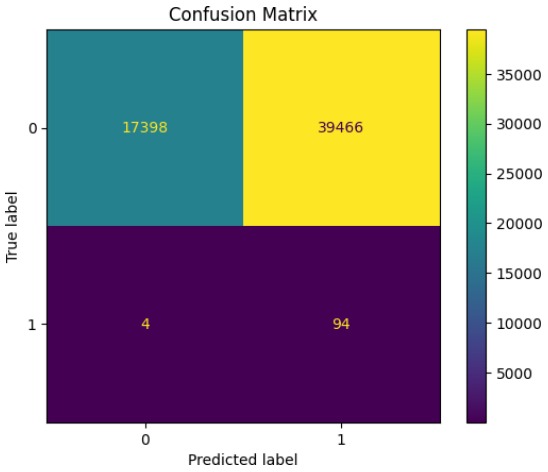
Random Forest Classifier model-



Testing Accuracy: 0.23821143920508409

	precision	recall	f1-score	support
0	1.00	0.24	0.38	56864
1	0.00	0.96	0.00	98
accuracy			0.24	56962
macro avg	0.50	0.60	0.19	56962
weighted avg	1.00	0.24	0.38	56962

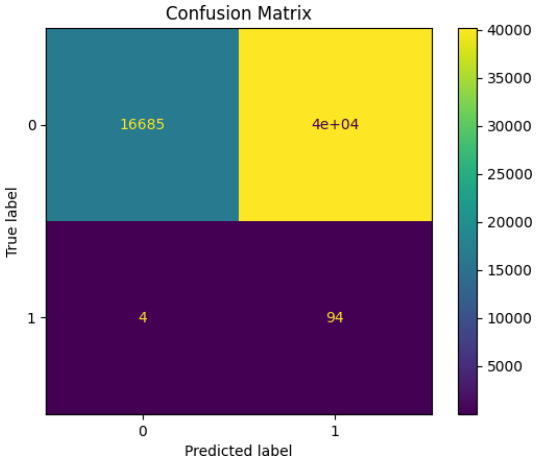
AdaBoost Classifier model-



Testing Accuracy: 0.3070819142586286

	precision	recall	f1-score	support
0	1.00	0.31	0.47	56864
1	0.00	0.96	0.00	98
accuracy			0.31	56962
macro avg	0.50	0.63	0.24	56962
weighted avg	1.00	0.31	0.47	56962

XGBoost Classifier model-



Testing Accuracy: 0.2945647975843545

	precision	recall	f1-score	support
0	1.00	0.29	0.45	56864
1	0.00	0.96	0.00	98
accuracy			0.29	56962
macro avg	0.50	0.63	0.23	56962
weighted avg	1.00	0.29	0.45	56962

The precision score for class 0 (representing non-fraudulent transactions) is consistently 100% across all models, indicating that when the model predicts a transaction as non-fraudulent, it is accurate 100% of the time.

However, there are notable differences in the recall scores. For the XGBoost model, the recall is 29%, for AdaBoost it's 31%, and for Random Forest, it's 24%. This suggests that these models are missing a substantial portion of actual non-fraudulent transactions. In contrast, the logistic regression (LR) model achieves a recall of 96%, while K-Nearest Neighbors (KNN) achieve a recall of 87%, indicating better performance in capturing non-fraudulent transactions.

The precision score for class 1 (representing fraudulent transactions) is nearly 0% across all models, suggesting that the models do not effectively identify any transactions as fraudulent. However, the recall is notably higher, with scores of 96% for XGBoost, AdaBoost and Random Forest, and 91% for KNN, and 89% for LR. This indicates that the models successfully detect nearly all actual fraudulent transactions.

The XGBoost, AdaBoost, and Random Forest models exhibit higher f1-scores for non-fraudulent transactions compared to fraudulent ones, with values ranging from 0.38 to 0.47 for class 0 and 0.00 for class 1. For class 0, the f1-score values range from 93% to 98%, with the Logistic Regression model achieving the highest f1-score, followed by K-Nearest Neighbors models. However, for class 1, the f1-score values are considerably lower, ranging from 2% to 7% across these models, highlighting difficulties in correctly identifying fraudulent transactions.

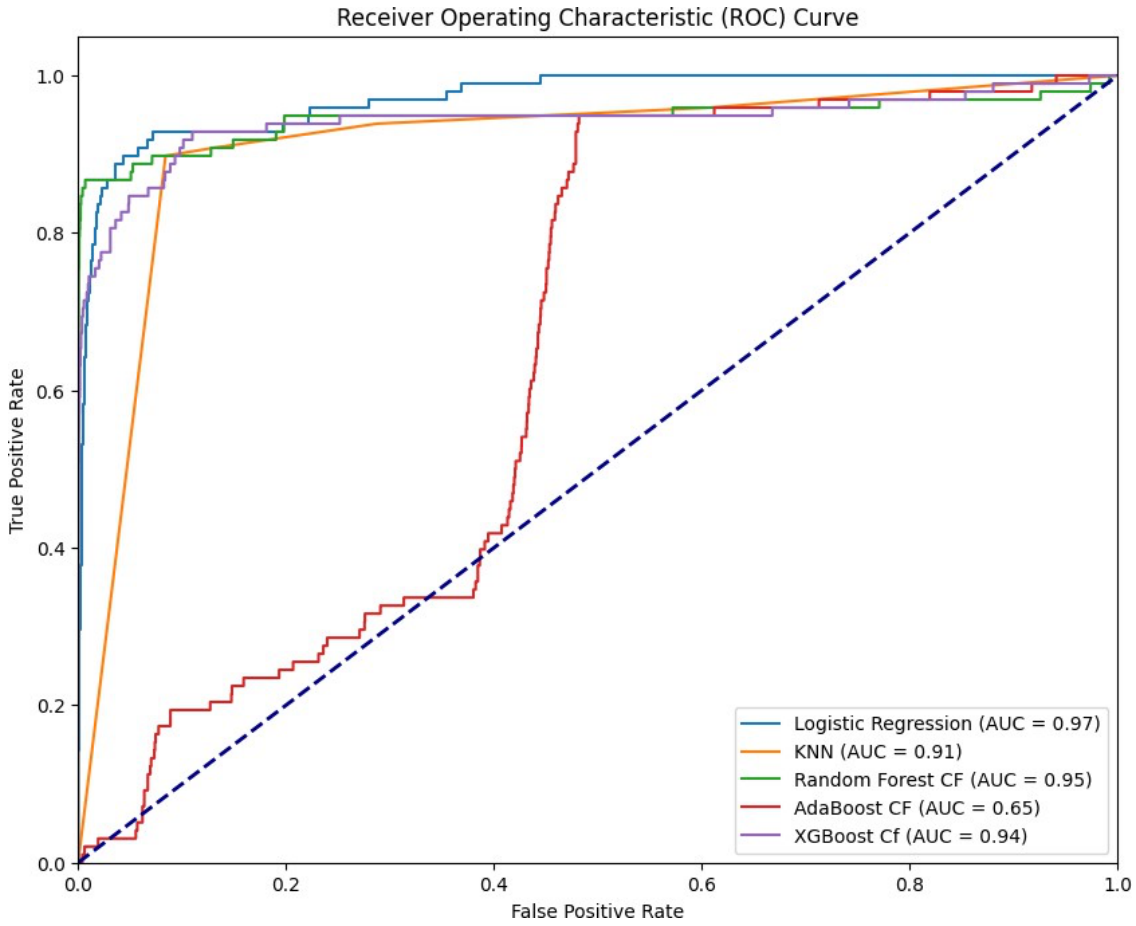
The testing accuracies for various models are as follows:

- Logistic Regression: 95.75%
- K-Nearest Neighbors (KNN): 86.91%
- Random Forest: 23.82%
- AdaBoost: 30.71%
- XGBoost: 29.46%

These values represent the percentage of correctly predicted outcomes on the testing dataset for each respective model. Among these models, Logistic Regression demonstrates the highest testing accuracy, followed by K-Nearest Neighbors. Conversely, Random Forest exhibits notably lower testing accuracy compared to the other models.

Overall, the model shows high recall for fraudulent transactions but poor precision and f1-score, indicating that it is better at identifying fraud but struggles with accuracy in generalizing predictions.

The results below indicate the Area Under the Receiver Operating Characteristic Curve (ROC AUC) for each model.



- Logistic Regression has an AUC of 0.97, indicating that it performs very well in terms of separating the classes, with a high true positive rate and a low false positive rate.
- K-Nearest Neighbors (KNN) have an AUC of 0.91, suggesting they also perform well, but not as well as Logistic Regression.
- Random Forest and XGBoost have an AUC of 0.95, indicating strong performance similar to Logistic Regression.
- AdaBoost has the lowest AUC of 0.65, implying that it performs relatively poorly compared to the other models in distinguishing between classes.

In a ROC AUC plot, higher AUC values correspond to better model performance, with the curve being closer to the top-left corner of the plot. Therefore, the results suggest that Logistic Regression, Random Forest, and XGBoost are the top-performing models, while AdaBoost lags behind in terms of class separation ability.

The NearMiss undersampling technique did not perform well in our credit card fraud detection scenario for the following reasons:

- **Loss of Information:** NearMiss aggressively removes majority class samples that are close to minority class samples, leading to a significant loss of valuable information. This can hinder the ability of models to learn complex relationships present in the data.
- **Model Complexity:** Ensemble methods like Random Forest, AdaBoost, and XGBoost are adept at capturing complex patterns in data. However, when trained on a drastically reduced dataset due to undersampling, these models might struggle to learn effectively, resulting in suboptimal performance.
- **Vulnerability to Noise:** NearMiss may inadvertently retain noisy or irrelevant minority class samples while removing informative majority class samples. This can introduce noise into the training data, negatively impacting the performance of ensemble methods sensitive to data quality.
- **Imbalanced Class Distribution:** Despite attempting to balance the class distribution, NearMiss may still leave the dataset imbalanced, with the minority class underrepresented. Ensemble methods may struggle to learn from such imbalanced data, leading to biased predictions and reduced performance.

Performing Over Sampling : SMOTE (Synthetic Minority Over-sampling Technique)

We have used the SMOTETomek method here as it can be an effective approach for improving classification accuracy and mitigating the impact of class imbalance.

SMOTETomek is a hybrid resampling technique that combines the over-sampling method Synthetic Minority Over-sampling Technique (SMOTE) with the under-sampling method Tomek Links.

Here's a brief overview of how SMOTETomek works:

- **SMOTE (Synthetic Minority Over-sampling Technique):** This technique generates synthetic samples for the minority class by interpolating new instances between existing minority class samples. It helps address the class imbalance by increasing the number of minority class samples.
- **Tomek Links:** Tomek Links are pairs of instances from different classes that are nearest neighbors of each other. By removing the majority class instances from these pairs, Tomek Links can help clarify the decision boundary between classes and potentially improve the performance of classifiers.
- **Combination:** SMOTETomek combines the strengths of SMOTE and Tomek Links. First, it oversamples the minority class using SMOTE to increase its representation. Then, it undersamples both the majority and minority classes using Tomek Links to remove redundant and noisy samples, thereby improving the balance and quality of the dataset.

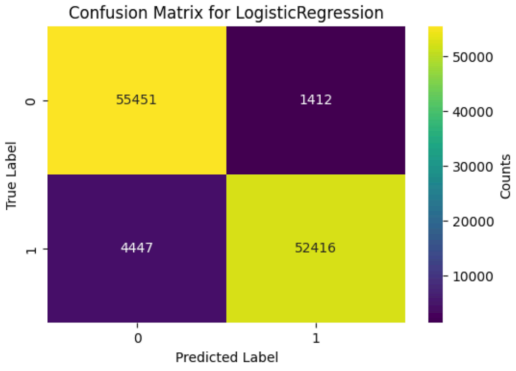
To address the severe class imbalance in our dataset, we applied the SMOTETomek method with a sampling strategy of 0.75 on our training dataset. Before applying the SMOTETomek method, the class distribution was heavily skewed, with the majority class (Non-fraudulent Transactions) dominating the dataset with 227,451 instances, while the minority class (Fraudulent Transactions) had only 394 instances.

After applying the SMOTETomek method, the class distribution was significantly improved, with the number of instances in the minority class increased to 170,588, resulting in a more balanced dataset overall.

Results of Models trained on Oversampled data:

Following oversampling of the dataset, various classification models were trained using the balanced data. These models include Logistic Regression, K-Nearest Neighbors (KNN), Random Forest Classifier, as well as boosting models like AdaBoost and XGBoost. The results of these models are presented below.

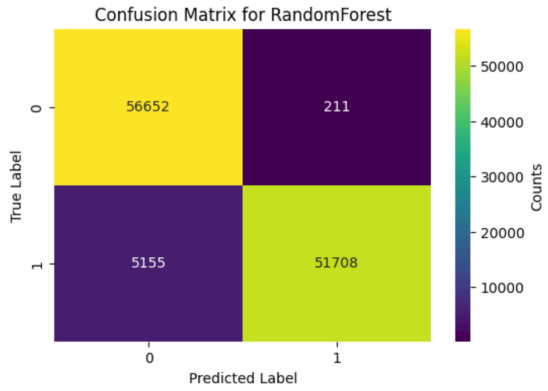
Logistic Regression model-



	precision	recall	f1-score	support
0	0.93	0.98	0.95	56863
1	0.97	0.92	0.95	56863
accuracy			0.95	113726
macro avg	0.95	0.95	0.95	113726
weighted avg	0.95	0.95	0.95	113726

Testing Accuracy for LogisticRegression: 0.9484814378418304
ROC-AUC Score for LogisticRegression: 0.9895110448957775

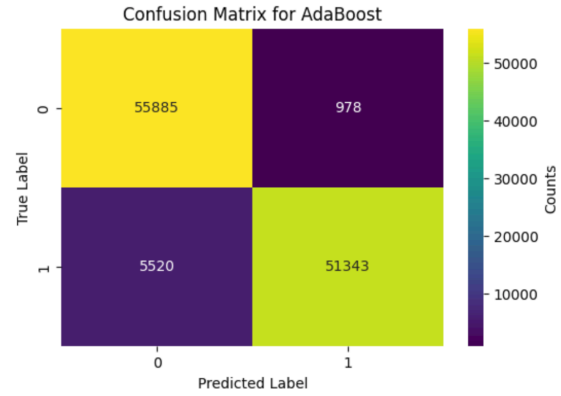
Random Forest Classifier model-



	precision	recall	f1-score	support
0	0.92	1.00	0.95	56863
1	1.00	0.91	0.95	56863
accuracy			0.95	113726
macro avg	0.96	0.95	0.95	113726
weighted avg	0.96	0.95	0.95	113726

Testing Accuracy for RandomForest: 0.9528164184091589
ROC-AUC Score for RandomForest: 0.9941903115815087

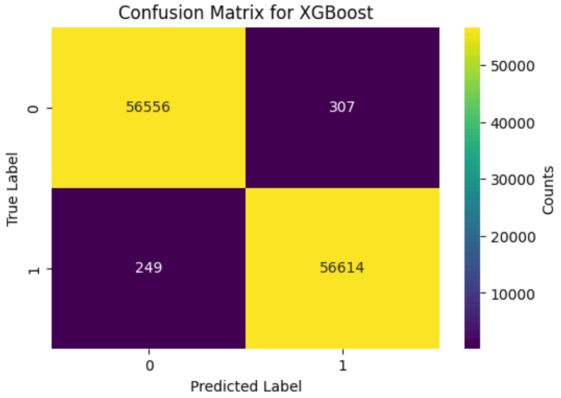
AdaBoost Classifier model-



	precision	recall	f1-score	support
0	0.91	0.98	0.95	56863
1	0.98	0.90	0.94	56863
accuracy			0.94	113726
macro avg	0.95	0.94	0.94	113726
weighted avg	0.95	0.94	0.94	113726

Testing Accuracy for AdaBoost: 0.9428626699259623
ROC-AUC Score for AdaBoost: 0.990994204838701

XGBoost Classifier model-



Classification Report for XGBoost:

	precision	recall	f1-score	support
0	1.00	0.99	1.00	56863
1	0.99	1.00	1.00	56863
accuracy			1.00	113726
macro avg	1.00	1.00	1.00	113726
weighted avg	1.00	1.00	1.00	113726

Testing Accuracy for XGBoost: 0.9951110563987127
ROC-AUC Score for XGBoost: 0.9997772238731102

The precision score for class 0 (representing non-fraudulent transactions) is consistently high across all models, indicating that the models are able to correctly identify non-fraudulent transactions most of the time. For all models, the precision for class 0 is between 91% and 100%.

However, there are slight differences in the recall scores for the models when detecting class 1 (fraudulent transactions). The **XGBoost** and **Random Forest** models achieve the highest recall at 100%, while **AdaBoost** follows closely with 90%. The **Logistic Regression** model has the lowest recall for fraudulent transactions at 92%. These recall scores indicate how well the models capture fraudulent transactions.

When considering precision for class 1 (fraudulent transactions), **XGBoost** outperforms the other models with a precision score of 99%, followed by **Logistic Regression** at 97%, **Random Forest** at 91%, and **AdaBoost** at 98%. These precision scores indicate how well the models correctly classify fraudulent transactions.

The f1-score, which combines both precision and recall, is highest for **XGBoost** at 99%, followed by **Random Forest** and **Logistic Regression** with scores around 95%, and **AdaBoost** at 94%. This shows that all models perform well, but **XGBoost** provides the best overall balance between precision and recall for identifying fraudulent transactions.

All models exhibit higher f1-scores for non-fraudulent transactions compared to fraudulent transactions, with values ranging from 94% to 100% for class 0 and 90% to 100% for class 1. The **XGBoost** model achieves the highest f1-score of 100% for both classes, indicating near-perfect classification ability.

The testing accuracy for various models is as follows:

- **Logistic Regression**: 94.84%
- **Random Forest**: 95.28%
- **AdaBoost**: 94.28%
- **XGBoost**: 99.51%

These values represent the percentage of correctly predicted outcomes on the testing dataset and show that the models perform well. Among these models, **XGBoost** demonstrates the highest overall accuracy, followed by **Random Forest**, **AdaBoost**, and **Logistic Regression**.

Model-Specific Observations:

1. **Logistic Regression**:
- Testing Accuracy: 94.84%
- ROC-AUC Score: 0.9895

- Precision, recall, and f1-score for non-fraudulent transactions are all high, with a slight drop in recall for fraudulent transactions.

2. **Random Forest**:

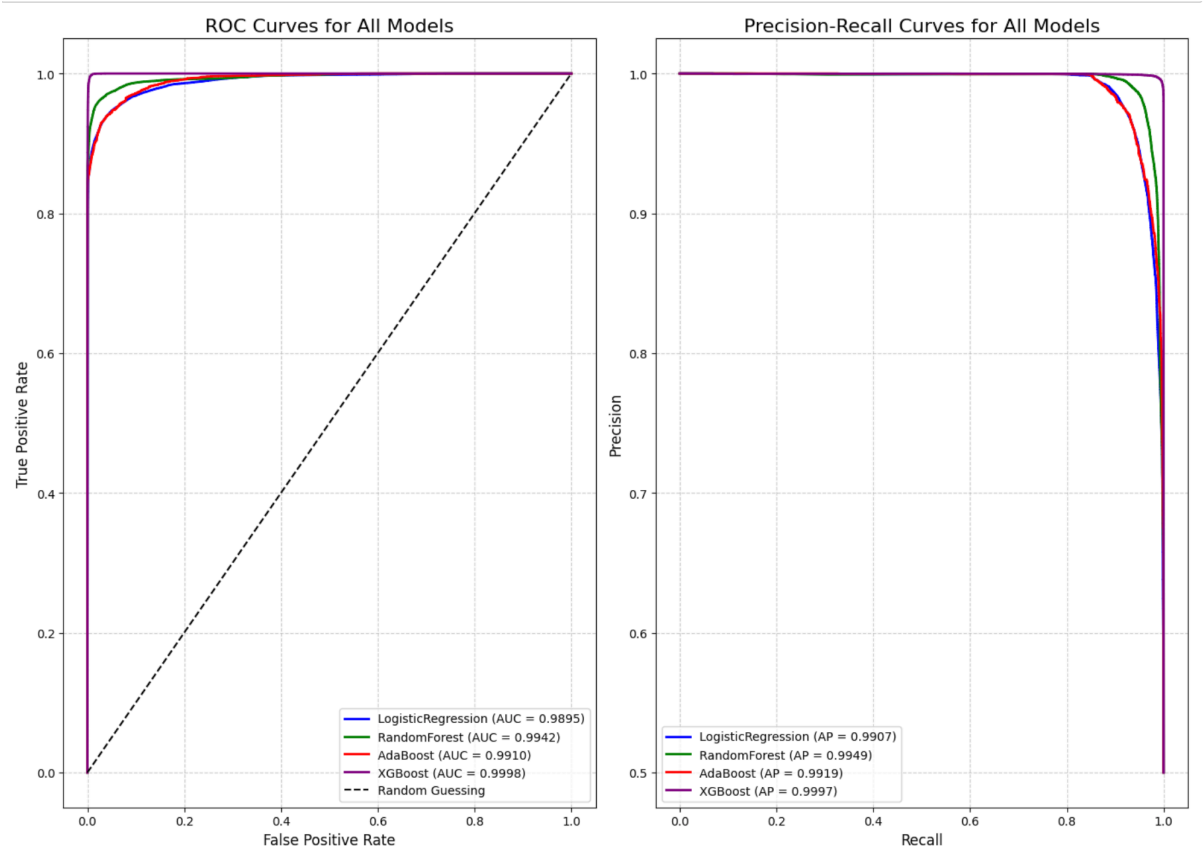
- Testing Accuracy: 95.28%
- ROC-AUC Score: 0.9942
- Achieves perfect recall for non-fraudulent transactions, with strong precision for fraudulent transactions.

3. **AdaBoost**:

- Testing Accuracy: 94.28%
- ROC-AUC Score: 0.9909
- High recall for fraudulent transactions, but slightly lower precision compared to **XGBoost**.

4. **XGBoost**:

- Testing Accuracy: 99.51%
- ROC-AUC Score: 0.9998
- Best performer across all metrics with near-perfect precision, recall, and f1-scores for both fraudulent and non-fraudulent transactions.



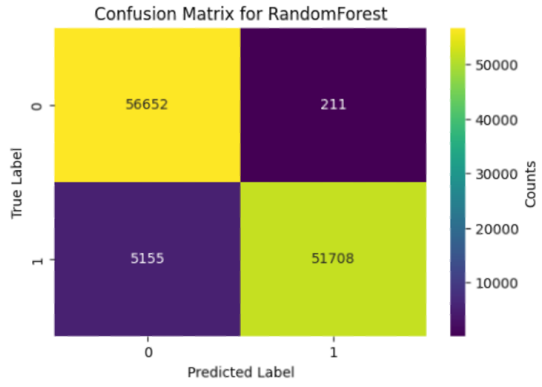
- All these models have an AUC > 0.90 (ranging 0.93 to 0.99) indicating good performance in distinguishing between classes.
- XGBoost and RandomForest CF has highest AUC of 0.99, showcasing that they have performed very well in terms of separating the classes, with a high true positive rate and a low false positive rate.

In a ROC AUC plot, higher AUC values correspond to better model performance, with the curve being closer to the top-left corner of the plot. Therefore, the results suggest that **XGBoost** and Random Forest are the top-performing models.

Hyperparameter Tuning of Random Forrest model to get its best performance:

Among the various models and balancing methods experimented with, the XGBoost model stands out as the top performer when using oversampling techniques. But for now we are using Random Forrest because it consumes less computational cost .We have further optimized its parameters through hyperparameter tuning, and identified the Random Forrest model that exhibits the best performance across various evaluation metrics including accuracy, precision, recall, and F1-score as follows-

Random Forest Classifier model-



	precision	recall	f1-score	support
0	0.92	1.00	0.95	56863
1	1.00	0.91	0.95	56863
accuracy			0.95	113726
macro avg	0.96	0.95	0.95	113726
weighted avg	0.96	0.95	0.95	113726

Testing Accuracy for RandomForest: 0.9528164184091589
ROC-AUC Score for RandomForest: 0.9941903115815087

Hyperparameter tuning involved systematically exploring different combinations of parameters to maximize the model's predictive power. The chosen model has undergone rigorous testing and validation to ensure its robustness and generalization to unseen data. We'll deploy this top-performing model as our final choice for deployment

CONCLUSION:

Our dataset exhibits significant class imbalance, with the majority of transactions being non-fraudulent (99.83%). This presents a challenge for predictive modeling, as algorithms may struggle to accurately detect fraudulent transactions amidst the overwhelming number of legitimate ones. To address this issue, we employed a combination of **undersampling** and **SMOTETomek** oversampling techniques to balance the class distribution and improve the models' ability to identify fraudulent transactions.

1. **Undersampling**:

Initially, we attempted to address class imbalance by applying the **NearMiss** undersampling technique, which reduced the number of non-fraudulent transactions to match the number of fraudulent transactions. However, this approach did not yield satisfactory results, as reducing the majority class led to a significant loss of valuable data, and the model struggled to generalize well due to the reduced dataset size.

2. **Oversampling with SMOTETomek**:

To further address the imbalance, we applied the **SMOTETomek** technique, which combines **SMOTE** to oversample the minority class (fraudulent transactions) and **Tomek links** to remove noisy data points from the majority class. This approach created a more balanced and representative dataset, enabling the models to better capture the patterns of fraudulent transactions.

3. **Machine Learning Models**:

After preprocessing and balancing the dataset, we trained several machine learning models, including:

- **Logistic Regression**
- **Random Forest Classifier**
- **AdaBoost Classifier**
- **XGBoost Classifier**

4. **Evaluation Metrics**:

We evaluated the performance of each model using various metrics such as accuracy, precision, recall, F1-score, and ROC-AUC. Additionally, hyperparameter tuning was applied to improve model performance.

5. **Model Selection**:

Although **XGBoost** demonstrated the best performance across several metrics, including precision, recall, and ROC-AUC, I ultimately selected **Random Forest** for hyperparameter tuning. This decision was made to simplify the tuning process while still maintaining strong model performance. **Random Forest** provided robust results, achieving high accuracy and balanced performance in detecting fraudulent transactions. After hyperparameter tuning, the **Random Forest** model presented a solid trade-off between performance and simplicity, making it the most practical choice for this project.

FUTURE WORK:

1. Anomaly Detection Techniques

- **Isolation Forests** and **Autoencoders** could be integrated to detect anomalies within the dataset. These techniques are particularly useful in identifying outliers that deviate from normal behaviour, which can help flag potential fraud cases that traditional classification models might miss.
- **Autoencoders**, for instance, can be used to reconstruct input data and detect deviations as anomalies, which can add another layer of precision to the detection process.

2. Advanced Deep Learning Models

- Implementing deep learning architectures such as **Convolutional Neural Networks (CNNs)** and **Recurrent Neural Networks (RNNs)** offers strong potential for processing large-scale, structured data, such as credit card transactions.
- **CNNs** can be explored for feature extraction, while **RNNs** are effective in handling sequential data, which could be used to identify temporal patterns in transaction data indicative of fraudulent behaviour.

3. Hybrid Models

- Combining deep learning models with traditional machine learning approaches, such as **Random Forests** and **XGBoost**, could create a more powerful hybrid system. This approach would leverage the strengths of both methods to improve accuracy, adaptability, and model performance.

4. Unsupervised Learning and Transfer Learning

- **Unsupervised Learning** techniques could be used to identify fraud patterns without requiring labeled data, which could be helpful in detecting new types of fraud that have not been previously encountered.
- **Transfer Learning** could be used to enhance model performance by applying knowledge gained from one fraud detection dataset to another, thereby increasing efficiency and effectiveness when dealing with new data.

5. Real-Time Fraud Detection

- Exploring real-time fraud detection using streaming technologies, such as **Apache Kafka** or **Apache Flink**, could enable the model to analyse transactions as they happen. This would help financial institutions respond to fraud in real-time, reducing the risk of financial loss.

6. Continuous Learning Systems

- Developing a system for **continuous learning** could allow the model to evolve over time, adapting to new fraud patterns as they emerge. This would ensure that the model remains effective in the face of changing fraud strategies.