# FindDefault-Prediction-of-Credit-Card-fraud

## Problem Statement:

Credit cards are widely used for online transactions, offering convenience but also exposing users to the risk of fraud. **Credit card fraud** refers to the unauthorized use of someone else’s credit card details to make purchases or withdrawals. It is essential for credit card companies to effectively identify fraudulent transactions to prevent customers from being charged for unauthorized purchases. This project focuses on building a classification model to predict whether a transaction is fraudulent or not based on a dataset containing credit card transactions from European cardholders in September 2013.

The dataset includes 284,807 transactions, of which 492 are fraudulent, making the dataset highly imbalanced. The positive class (fraudulent transactions) accounts for just 0.17% of all transactions.

## Data Overview:

The dataset consists of 284,807 rows and 31 columns. To protect the privacy of users, the dataset provider applied **Principal Component Analysis (PCA)** to transform the original numerical features into 28 principal components, along with the columns **Time**, **Amount**, and **Class**. The **Class** column is the target variable, with values indicating fraudulent (1) and non-fraudulent (0) transactions.

## Data Cleaning:

Upon inspecting the data for null and duplicate values, we found that the dataset contained no null values. However, it did include some duplicates, which were removed before proceeding with the analysis.

## Exploratory Data Analysis (EDA):

We explored the dataset to understand the distribution of fraudulent and non-fraudulent transactions. A graph was plotted to visualize this distribution. We found that fraudulent transactions account for just **0.17%** of the total dataset. Additionally, we examined the **Time** and **Amount** columns, plotting graphs to determine when most transactions took place and whether there were any outliers in the **Amount** column.

# Feature Engineering:

Since the **Time** and **Amount** columns were deemed unnecessary for modeling, they were removed. A new column, **scaled values**, was added to represent the scaled **Amount** feature. For the purpose of model training, the data was split into two variables:

- **X**: The feature matrix containing the PCA components and the scaled **Amount**.
- **Y**: The target variable, representing the **Class**.

# Model Training:

We split the dataset into training and testing sets using a **70-30** ratio via the train_test_split() function, with the following parameters:

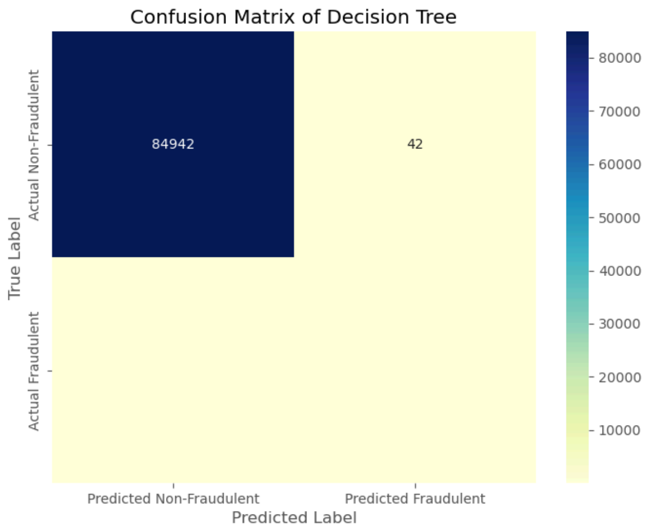
- **X**: Feature matrix
- **Y**: Target variable
- **test_size**: 0.3 (30% of the data was allocated to the test set)

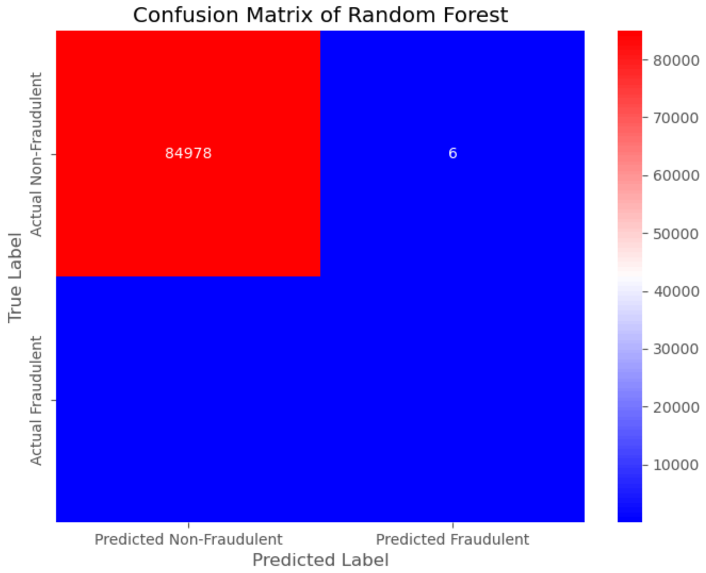
# Model Selection:

We selected two machine learning algorithms for model building:

- **Decision Tree**: A supervised learning algorithm that splits the data into subsets based on feature values to form a tree-like structure of decisions.
- **Random Forest**: An ensemble learning method that combines multiple decision trees to improve accuracy and reduce overfitting.

We evaluated the performance of each algorithm to determine which one performs better for our dataset.





# Model Validation:

We validated and tested both models using various evaluation metrics, including:

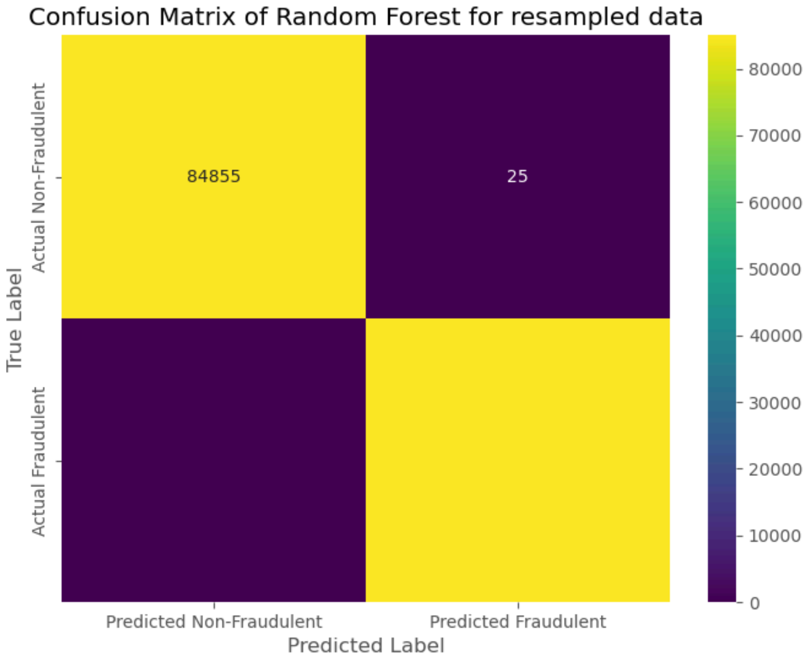
- **Accuracy Score**
- **Precision Score**
- **Recall Score**
- **F1 Score**
- **Confusion Matrix**

Heatmaps were generated for the confusion matrices of both models to visually assess their performance.

# Dealing with Imbalanced Data:

The dataset is highly imbalanced, with fraudulent transactions making up only 0.17% of the total. To address this, we applied the **SMOTE (Synthetic Minority Oversampling Technique)**, which is an oversampling method designed to balance the class distribution by generating synthetic examples of the minority class.

After balancing the data, we trained the **Random Forest** model on the resampled dataset, as **Random Forest** had shown better performance than **Decision Tree**. We then evaluated the model using the same performance metrics (accuracy, precision, recall, F1 score, confusion matrix) and generated heatmaps for the confusion matrices.



# Model Deployment:

To prepare for model deployment, we plan to use the **pickle** library to save both the trained model and the dataframe for future use. This will allow for easy integration into production environments where predictions can be made on new, unseen data.

# Conclusion:

The model demonstrated improved performance after addressing the class imbalance using SMOTE and training with the **Random Forest** algorithm. With an accuracy exceeding 99%, the model shows strong potential for identifying fraudulent transactions in real-world applications. Moving forward, we aim to deploy the model and continuously monitor its performance to ensure its effectiveness.