Credit Card Fraud Prediction

Problem Statement

A credit card is one of the most used financial products to make online purchases and payments. Though the Credit cards can be a convenient way to manage your finances, they can also be risky. Credit card fraud is the unauthorized use of someone else's credit card or credit card information to make purchases or withdraw cash.

It is important that credit card companies are able to recognize fraudulent credit card transactions so that customers are not charged for items that they did not purchase.

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.

We have to build a classification model to predict whether a transaction is fraudulent or not.

Contents

Data Summary

- Total Transactions: 284,807

- Fraudulent Transactions: 492 (0.172% of all transactions)

Exploratory Data Analysis

. 1.Distribution of Transaction Amounts:

- The majority of transactions have small amounts.

- There are a few transactions with very large amounts, indicating a highly skewed distribution.

2. Distribution of Transaction Times:

- Transactions are spread across the two days.

- There are peaks and troughs, suggesting varying transaction frequencies at different times.

3. Correlation Matrix Heatmap:

- Most features (V1 to V28) have very low correlation with each other and with the 'Amount' and 'Time' features.

- There are no highly correlated features, suggesting that PCA effectively reduced multicollinearity.

4. Distribution of Class Labels:

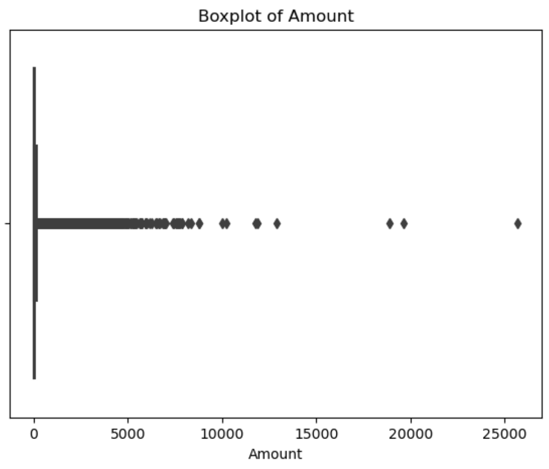
- The dataset is highly imbalanced with a significant majority of transactions being non-fraudulent (Class 0).

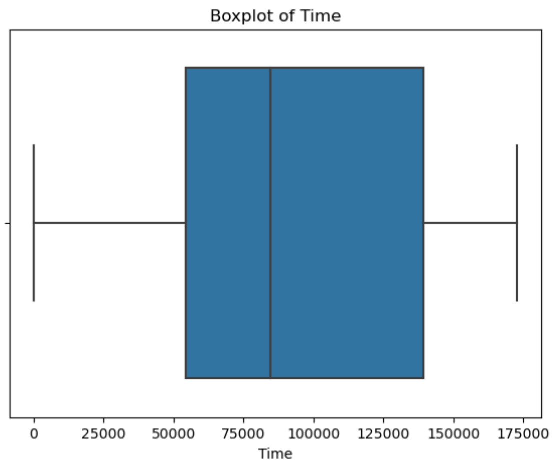
- Fraudulent transactions (Class 1) are very few in comparison, constituting only 0.172% of the total.

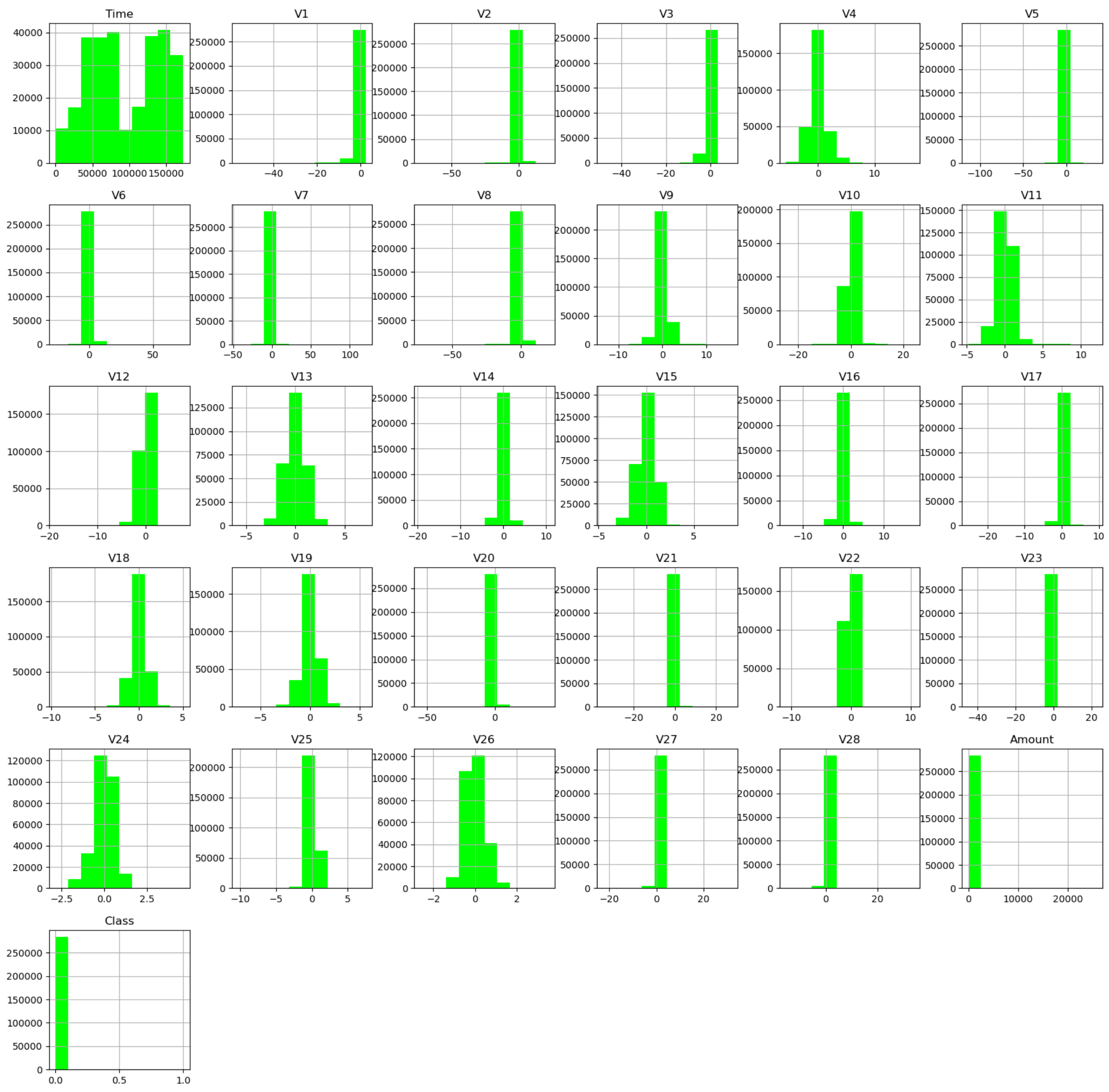
- The dataset is highly imbalanced, and fraud detection models will need to account for this imbalance.

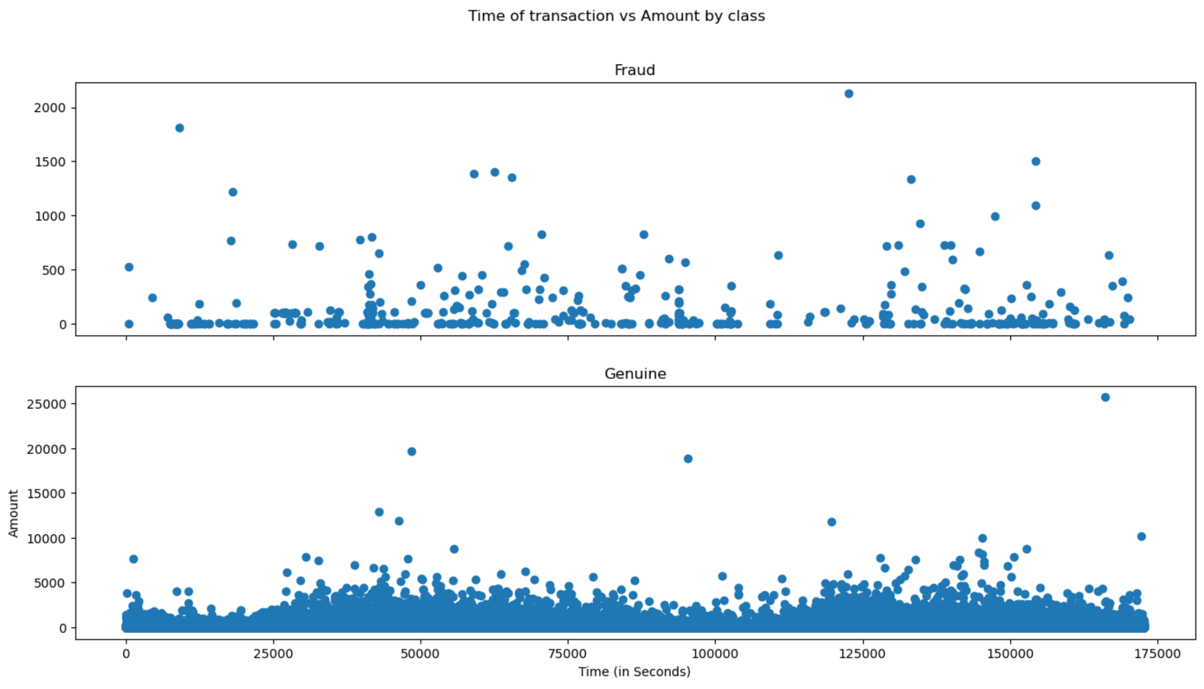
- Feature correlations are low, indicating the PCA features are likely orthogonal.

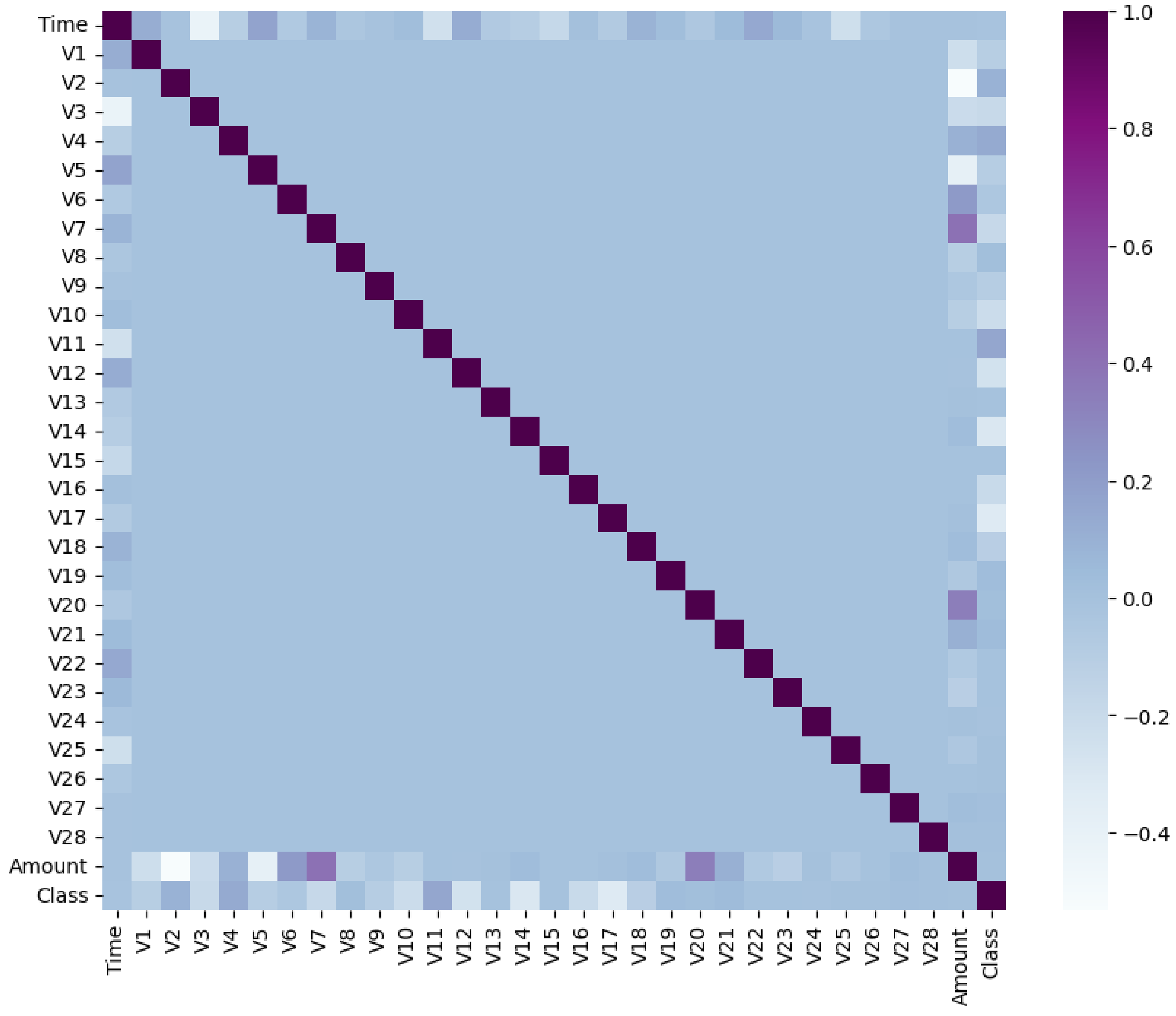
- The high skewness in transaction amounts suggests that feature scaling or transformation may be necessary before model training






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


Time of transaction vs Amount by class
Fraud
Genuine
Amount
Time (in Seconds)
0
500
1000
1500
2000
5000
10000
15000
20000
25000
25000
50000
75000
100000
125000
150000
175000


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

Data Cleaning

To clean the credit card transaction dataset, we need to perform several steps:

1. Check for Missing Values: There are no missing values in this dataset.

2. Handle Outliers: Identify and address outliers in the Amount and Time features.

3. Standardize Features: Standardize the Amount feature for better model performance.

4. Balance the Dataset

Dealing with Imbalanced data

Under-Sampling

Build a sample dataset containing similar distribution of normal transactions and Fraudulent Transactions

Number of Fraudulent Transactions --> 492

Feature Engineering

1. Handling Imbalance:

   - Techniques such as oversampling the minority class or undersampling the majority class.

   - Alternatively, using algorithms that are robust to class imbalance.

2. Feature Engineering:

   - Creating new features if necessary.

   - Scaling and transforming features where needed.

3. Feature Selection:

   - Identifying the most relevant features for the classification task.

   - Techniques like feature importance from tree-based models or recursive feature elimination (RFE).

Steps to Perform

1. Handling Class Imbalance:

   - Use the Synthetic Minority Over-sampling Technique (SMOTE) to balance the classes.

2. Scaling Features:

   - Standardizing the 'Amount' feature.

3. Feature Selection:

   - Using a tree-based classifier to determine feature importance.

Model Selection

1. Logistic Regression:

   - Simple and interpretable.

   - Works well with imbalanced data when combined with techniques like class weighting or resampling.

2. Decision Trees and Random Forests:

   - Can handle imbalanced data by adjusting the class weights.

- Random forests are robust and can capture non-linear relationships.

3. Gradient Boosting Machines (GBM):

   - Models like XGBoost, LightGBM, or CatBoost are powerful for handling imbalanced datasets.

   - They offer parameters to handle class imbalance effectively.

Model Training

To train models like Random Forest and XGBoost on the credit card fraud detection dataset, we need to follow a systematic approach. This includes preprocessing the data, splitting it into training and test sets, and training the models with hyperparameter tuning. Here's how you can do it step-by-step:

Step-by-Step Model Training

1. Data Preprocessing:

   - Handle missing values.

   - Standardize numerical features.

   - Balance the dataset.

2. Split the Data:

   - Split the data into training and test sets.

3. Train the Models:

   - Train Random Forest and XGBoost models.

4. Hyperparameter Tuning:

- Use GridSearchCV for hyperparameter tuning.

5. Model Evaluation:

- Evaluate the models using appropriate metrics.

Model Validation

Model validation is essential to ensure that the machine learning model generalizes well to unseen data. After performing hyperparameter tuning and selecting the best model, we need to validate the model using various techniques such as cross-validation and evaluating key performance metrics.

Steps for Model Validation

1. Cross-Validation:

- Use k-fold cross-validation to evaluate the model's performance more reliably.

2. Evaluate Performance Metrics:

- Assess the model using metrics such as precision, recall, F1-score, and ROC-AUC.

3. Confusion Matrix:

- Visualize the confusion matrix to understand the types of errors the model

Model Deployment

Deploying a machine learning model involves several steps, from saving the trained model to setting up an API for making predictions in a production environment. Below are the key steps to deploy the Random Forest or XGBoost model for fraud detection.

Steps for Model Deployment

1. Save the Trained Model:

- Use libraries like joblib or pickle to save the trained model to disk.

2. Set Up a Web Framework:

   - Use a web framework like Flask or FastAPI to create an API for making predictions.

3. Create API Endpoints:

   - Set up endpoints for making predictions and possibly for model monitoring.

4. Containerize the Application:

   - Use Docker to containerize the application for easier deployment and scalability.

5. Deploy the Container:

   - Deploy the container to a cloud platform like AWS, Google Cloud, or Azure.