Credit Card Fraud Detection

1- problem definition

Credit card fraud detection is a major concern for financial institutions worldwide, as fraudulent transactions can cause significant financial losses and damage customer trust. The goal of this project is to develop a machine learning model that can accurately distinguish between legitimate and fraudulent credit card transactions. Given the highly imbalanced nature of the dataset, this project prioritizes techniques that effectively identify rare fraudulent transactions without compromising the accuracy of detecting legitimate transactions.

2-dataset

The Credit Card Fraud Detection Dataset is a labeled dataset commonly used to train and evaluate machine learning models for detecting fraudulent credit card transactions. Each transaction is represented as a data point with several anonymized features, allowing for detailed analysis and model training without revealing sensitive customer information.

Dataset Composition

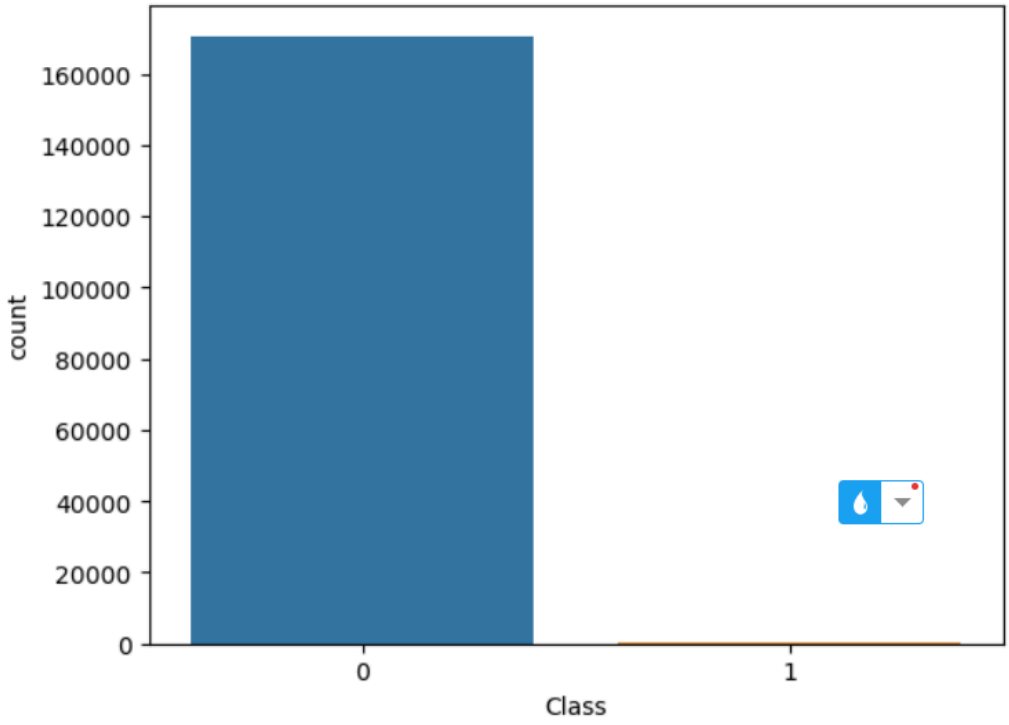
The dataset contains the following key columns:

1. **Time**:
    - Represents the seconds elapsed between the first transaction in the dataset and the current transaction.

    - This variable may capture patterns based on transaction time sequences but does not contain actual timestamps.
2. **V1 to V28 (Anonymized Features)**:
    - These 28 features are the result of Principal Component Analysis (PCA) transformation applied to the original dataset to protect user privacy.
    - These components are intended to capture the primary variation in the data and often include information about transaction patterns, behavior, and anomalies that may correlate with fraudulent activities.
3. **Amount**:
    - The monetary value of the transaction.
    - The transaction amount is an untransformed feature and could be valuable in predicting fraud, as certain amount ranges or patterns may correlate with fraud likelihood.
4. **Class (Target Variable)**:
    - This binary variable indicates whether a transaction is **fraudulent** (`1`) or **legitimate** (`0`).
    - Since the dataset is highly imbalanced, the majority of transactions are legitimate, with only a small fraction labeled as fraudulent.

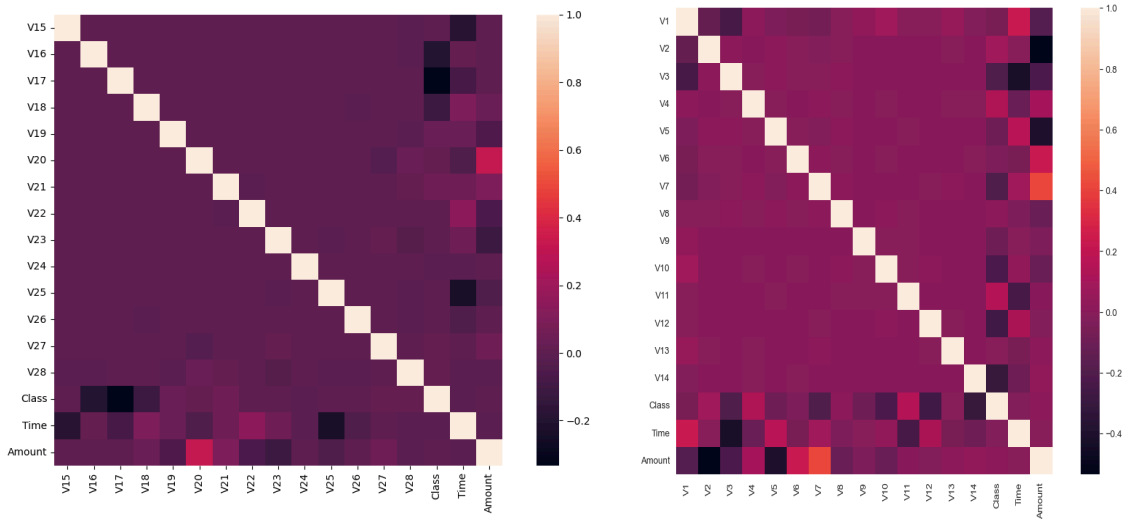
3-data analysis

3.1 target



here big impalned for class 0 so 0 is a majority class and 1 in minority class

3.2 -correlation between features



4-data preprocessing

Data Scaling:

Standardized the features in both the training and validation datasets using StandardScaler to ensure that all feature values have a mean of zero and a standard deviation of one. This improves model convergence and consistency by aligning the feature scales.

Data Resampling:

Addressed class imbalance by applying different resampling techniques to balance the dataset:
Random Undersampling: Reduced the number of non-fraudulent transactions, creating a smaller, balanced dataset.

SMOTE Oversampling: Increased the fraudulent samples by synthesizing new data points, enhancing the model's ability to detect rare fraud cases.

Combined Undersampling and Oversampling: First undersampled the majority class, then applied SMOTE to the minority class to achieve a balanced dataset without overloading it with synthetic data.

6-models

In this project, several machine learning models were trained and evaluated to determine the best-performing model for fraud detection. The following models were chosen based on their ability to handle imbalanced data and to achieve high F1 scores, a critical metric for fraud detection.

Models Used

1. **Logistic Regression**:
    - This linear model is straightforward and computationally efficient, making it a strong baseline. Logistic Regression works well with imbalanced datasets, especially when optimized for class separation.
2. **Random Forest Classifier**:
    - Random Forest is an ensemble method that combines multiple decision trees to improve prediction stability and accuracy. It is robust to overfitting, particularly useful in fraud detection due to

its ability to capture complex interactions between features.

3. **MLP (Multi-Layer Perceptron) Classifier**:
    - An MLP is a type of neural network that can learn complex patterns in data, making it suitable for high-dimensional problems like fraud detection. The model can be prone to overfitting, so careful tuning was performed to optimize its performance.
4. **Voting Classifier**:
    - The Voting Classifier combines the predictions of the above models to leverage their individual strengths. By using a majority voting strategy, this model tends to achieve better overall performance, especially when individual classifiers complement each other's weaknesses.

Hyperparameter Tuning with Random Search

To further improve model performance, hyperparameter tuning was performed using **RandomizedSearchCV**. This method allows exploration of a broad parameter space in a time-efficient manner by randomly selecting hyperparameters and evaluating their impact on model performance.

1. **Logistic Regression**:
    - **Hyperparameters Tuned**: `C` (regularization strength) and `max_iter` (maximum iterations).
    - **Objective**: Minimizing regularization while allowing enough iterations for convergence to prevent overfitting.
2. **Random Forest Classifier**:

    - **Hyperparameters Tuned**: `n_estimators` (number of trees), `max_depth` (maximum depth of each tree), and `min_samples_split` (minimum number of samples required to split a node).
    - **Objective**: Achieving a balance between model complexity and generalizability by controlling tree depth and the number of splits.
3. **MLP Classifier**:
    - **Hyperparameters Tuned**: `hidden_layer_sizes` (number of neurons in hidden layers), `activation` (activation function), and `alpha` (regularization term).
    - **Objective**: Preventing overfitting by selecting appropriate network sizes and regularization values to control complexity.
4. **Voting Classifier**:
    - **Hyperparameters Tuned**: Parameters of individual classifiers within the ensemble.
    - **Objective**: Creating an optimal combination of models by fine-tuning each model in the ensemble to contribute effectively to the final prediction.

Each model's performance was evaluated using the F1 score on a sample of the dataset to reduce computation time. Based on the best parameters from Random Search, each model was retrained on the full training set, followed by validation to confirm performance improvements.

Best Hyperparameters Summary:

1. **Random Forest Classifier**:

    - **Best Hyperparameters**:
        - `n_estimators`: 100
        - `min_samples_split`: 2
        - `min_samples_leaf`: 1
        - `class_weight`: {0: 1, 1: 0.1}
    - **Notes**: The class weight adjustment was necessary to account for class imbalance, prioritizing detection of minority class instances.
2. **MLP (Multi-Layer Perceptron) Classifier**:
    - **Best Hyperparameters**:
        - `solver`: 'adam'
        - `max_iter`: 150
        - `hidden_layer_sizes`: (150,)
        - `alpha`: 0.0001
        - `activation`: 'relu'
    - **Notes**: The `relu` activation function was retained for better learning, and the hidden layer size of 150 was found to balance model complexity and training efficiency effectively.
3. **Logistic Regression**:
    - **Best Hyperparameters**:
        - `solver`: 'saga'
        - `penalty`: 'elasticnet'
        - `l1_ratio`: 0.9
        - `class_weight`: {0: 1, 1: 0.2}
        - `C`: 10

    - **Notes**: Elastic net regularization (combining L1 and L2) with a high `l1_ratio` was optimal, improving performance while maintaining some regularization to avoid overfitting. Class weights were adjusted for better handling of class imbalance.
4. **XGBoost**:
    - **Best Hyperparameters**:
        - `subsample`: 0.9
        - `n_estimators`: 150
        - `max_depth`: 6
        - `learning_rate`: 0.2
        - `colsample_bytree`: 0.9
    - **Notes**: A slightly higher learning rate and a moderate tree depth enabled effective training, and subsampling helped in reducing overfitting.
5. **Voting Classifier**:
    - **Best Hyperparameters**:
        - `weights`: [1, 3, 1]
    - **Notes**: Weighting the ensemble models gave higher importance to the strongest individual classifiers, improving the overall performance by balancing their predictions effectively.

6-Evaluations

This report summarizes the performance of the models used in the classification task, focusing on the F1 Score and Precision-Recall Area Under the Curve (PR AUC). The

evaluation metrics provided below are derived from the training and validation datasets.

1. Random Forest Classifier

- **Training Metrics**:
  - **F1 Score**: 0.9637
  - **PR AUC**: 0.9665
- **Validation Metrics**:
  - **F1 Score**: 0.8743
  - **PR AUC**: 0.8797
- **Notes**: The Random Forest classifier demonstrated a strong performance during training, achieving a high F1 score and PR AUC. However, there is a noticeable drop in both metrics on the validation set, indicating some overfitting.

2. MLP Classifier

- **Training Metrics**:
  - **F1 Score**: 0.9885
  - **PR AUC**: 0.9890
- **Validation Metrics**:
  - **F1 Score**: 0.8276
  - **PR AUC**: 0.8287
- **Notes**: The MLP classifier shows exceptional training performance with very high F1 and PR AUC scores. However, it underperforms significantly on the validation set, suggesting potential overfitting or difficulty generalizing to unseen data.

3. Logistic Regression

- **Training Metrics**:
    - **F1 Score**: 0.8723
    - **PR AUC**: 0.8901
- **Validation Metrics**:
    - **F1 Score**: 0.7929
    - **PR AUC**: 0.7965
- **Notes**: Logistic Regression performs reasonably well, with moderate scores on both training and validation sets. The drop in performance is less severe compared to other models, indicating better generalization capability.

4. Voting Classifier

- **Training Metrics**:
    - **F1 Score**: 0.9578
    - **PR AUC**: 0.9613
- **Validation Metrics**:
    - **F1 Score**: 0.8757
    - **PR AUC**: 0.8796
- **Notes**: The Voting Classifier exhibits strong training performance and maintains a competitive validation score, effectively combining predictions from multiple classifiers.

5. XGBoost

- **Validation Metrics**:
    - **F1 Score**: 0.8554
    - **PR AUC**: 0.8617
- **Notes**: XGBoost performed decently, with reasonable F1 and PR AUC scores on the validation set. This model may benefit from further hyperparameter tuning to enhance its performance.

7-best model and testing

- voting classifier is the best
- in last test give me
- testing F1 Score for VotingClassifier is : 0.8528
- testing PR AUC for VotingClassifier is : 0.8531
-