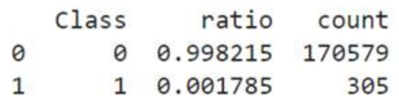
1 EDA Credit Fraud Detection

1.1 Introduction

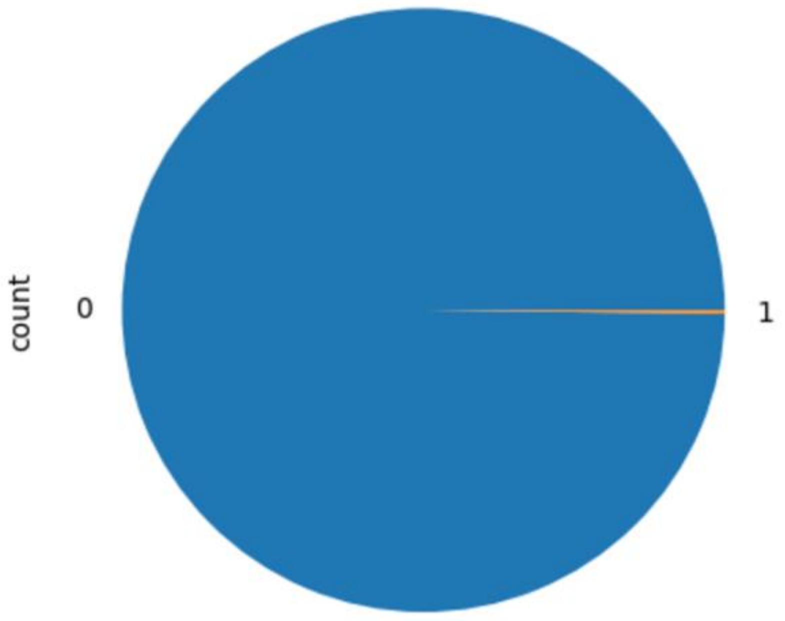
This project aims to build a model for detecting credit card fraud using a highly imbalanced dataset obtained from Kaggle. The dataset comprises 284,807 credit card transactions, of which only 492 are labeled as fraudulent. This extreme imbalance, where fraudulent transactions account for just 0.17% of the total, presents a significant challenge for conventional machine learning methods. To overcome this issue and enhance the detection of fraudulent activities, the project employs three strategies: a voting classifier, a neural network incorporating focal loss, and the Synthetic Minority Oversampling Technique (SMOTE).

1.2 Data Analysis

The dataset consists exclusively of numerical input variables, derived from a PCA transformation. Features labeled as V1, V2, ..., V28 represent the principal components obtained through PCA. The only features not subjected to PCA transformation are 'Time' and 'Amount.' The 'Time' feature records the seconds elapsed between each transaction and the first transaction in the dataset, while 'Amount' represents the transaction value. The 'Class' feature serves as the target variable, with a value of 1 indicating fraudulent transactions and 0 indicating non-fraudulent ones.

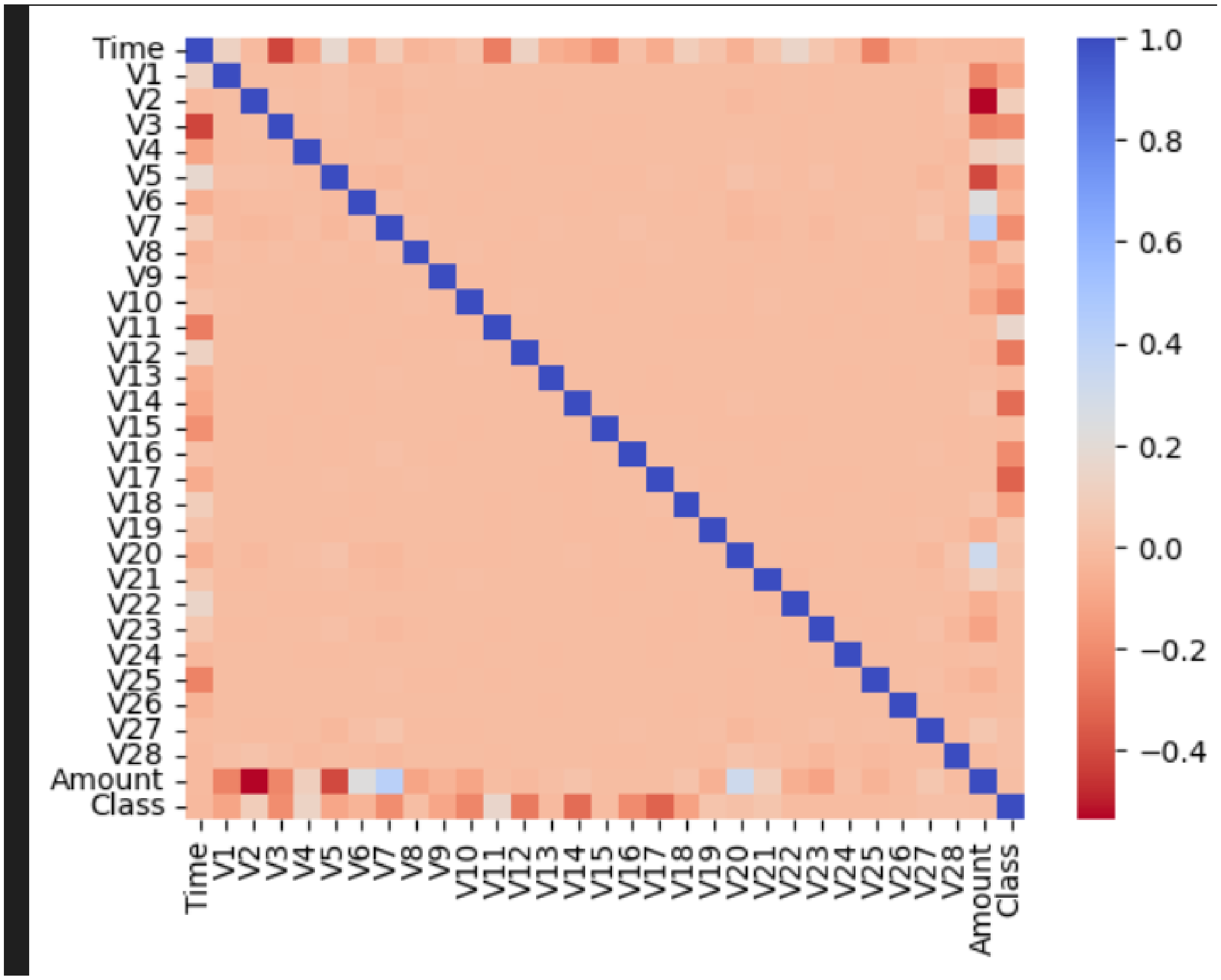


```
   Class     ratio   count
0      0  0.998215  170579
1      1  0.001785     305
```



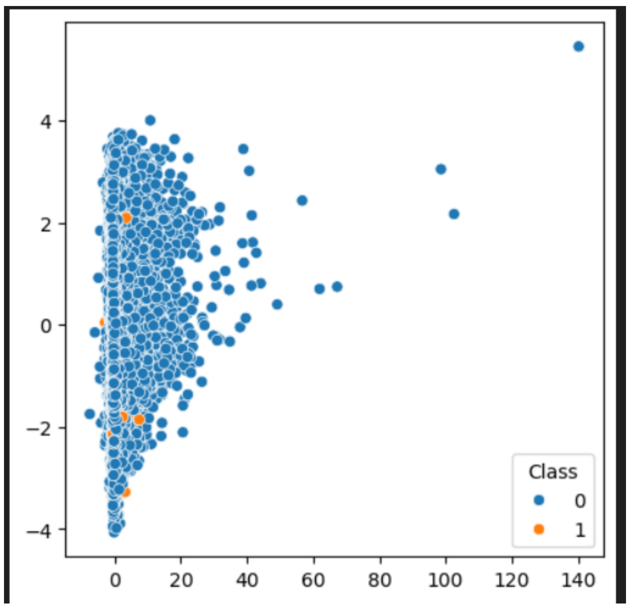
1.3 Data Correlation :

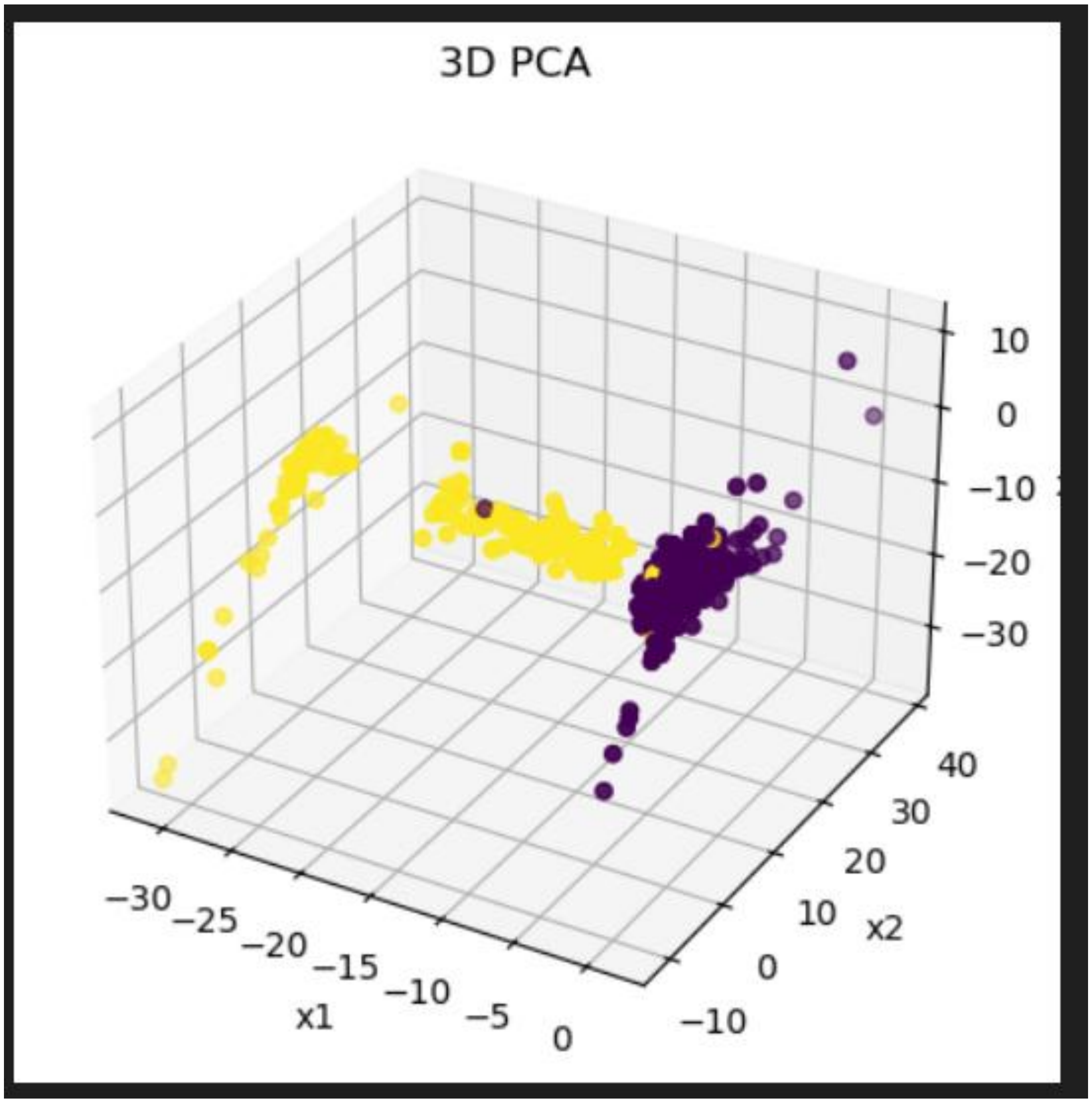
mostly weak correlations among variables. Most of the correlations are close to zero



1.4 Data Visualization Using PCA

2D





i think we could have good separable model here , I think they can have like a geometric relationship

1.5 Analysis of Random Forest

1. Most Important Features

V17 emerges as the most influential feature, contributing approximately 20% to the model's predictive power.

V14 follows, with a relative importance of around 17%.

V12 ranks third, accounting for nearly 14% of the importance.

2. Least Important Features

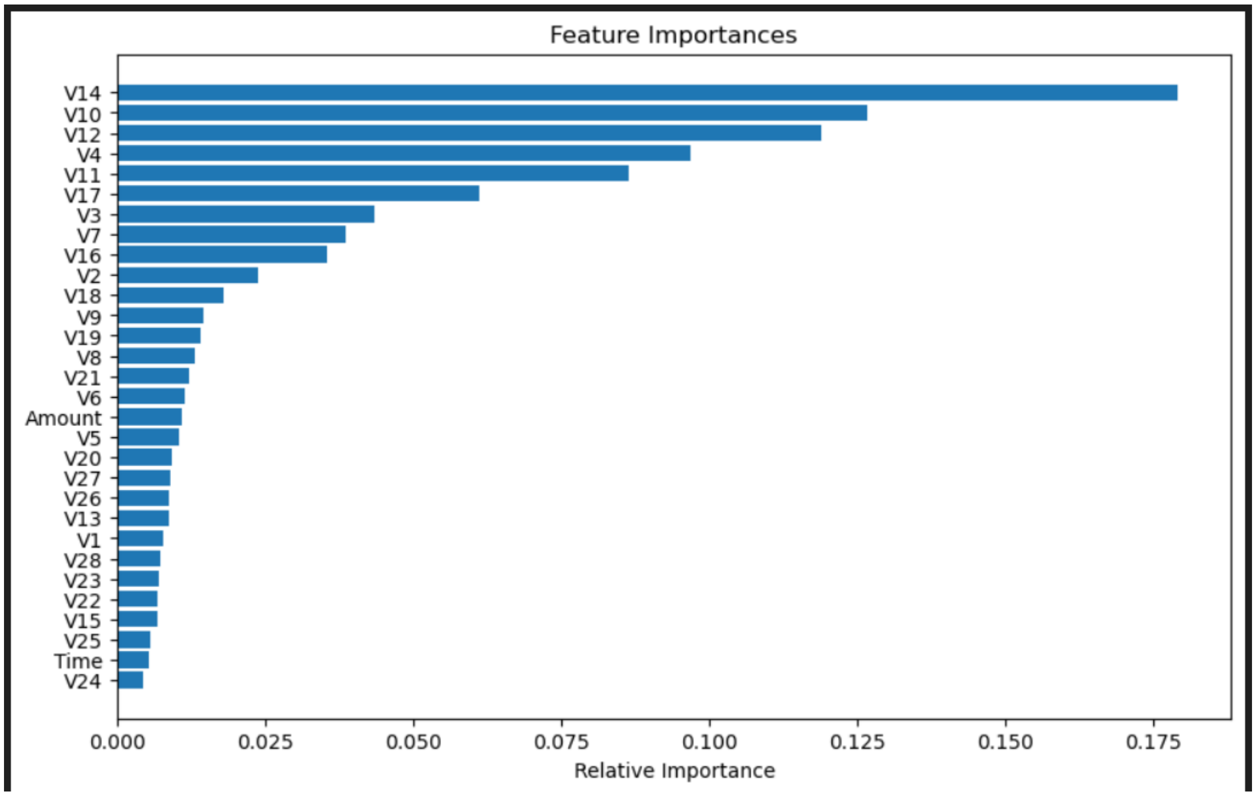
Several features, including V21, V1, V5, V6, V2, V19, V26, Amount, V13, V8, Time, V15, V20, V25, V22, V28, V24, V23, and V27, exhibit minimal importance, barely registering on the feature importance plot.

3. Key Observations

The 'Amount' feature shows unexpectedly low importance, suggesting that transaction value alone is not a strong indicator of fraudulent behavior in this model.

The 'Time' feature also has very low significance, implying that the timing of transactions does not play a critical role in fraud detection.

The features, labeled as V1, V2, etc., appear to be derived variables, likely principal components generated through a dimensionality reduction technique, such as PCA, prior to applying the Random Forest model.



2 Modeling & Results

Random Forest

A Randomized Search with Stratified K-Fold cross-validation was conducted for hyperparameter tuning.

Random Forest models showed a tendency to overfit the training data.

Regularization techniques were critical to improving generalization.

Increasing the minimum number of samples per leaf significantly reduced overfitting.

2. Logistic Regression

Initially, thiesmodel yielded subpar performance, but notable improvements were achieved through:

Scaling: The use of RobustScaler provided superior results compared to other scaling methods.

Class Weights: Setting the class_weight parameter significantly improved the model's ability to handle class imbalance.

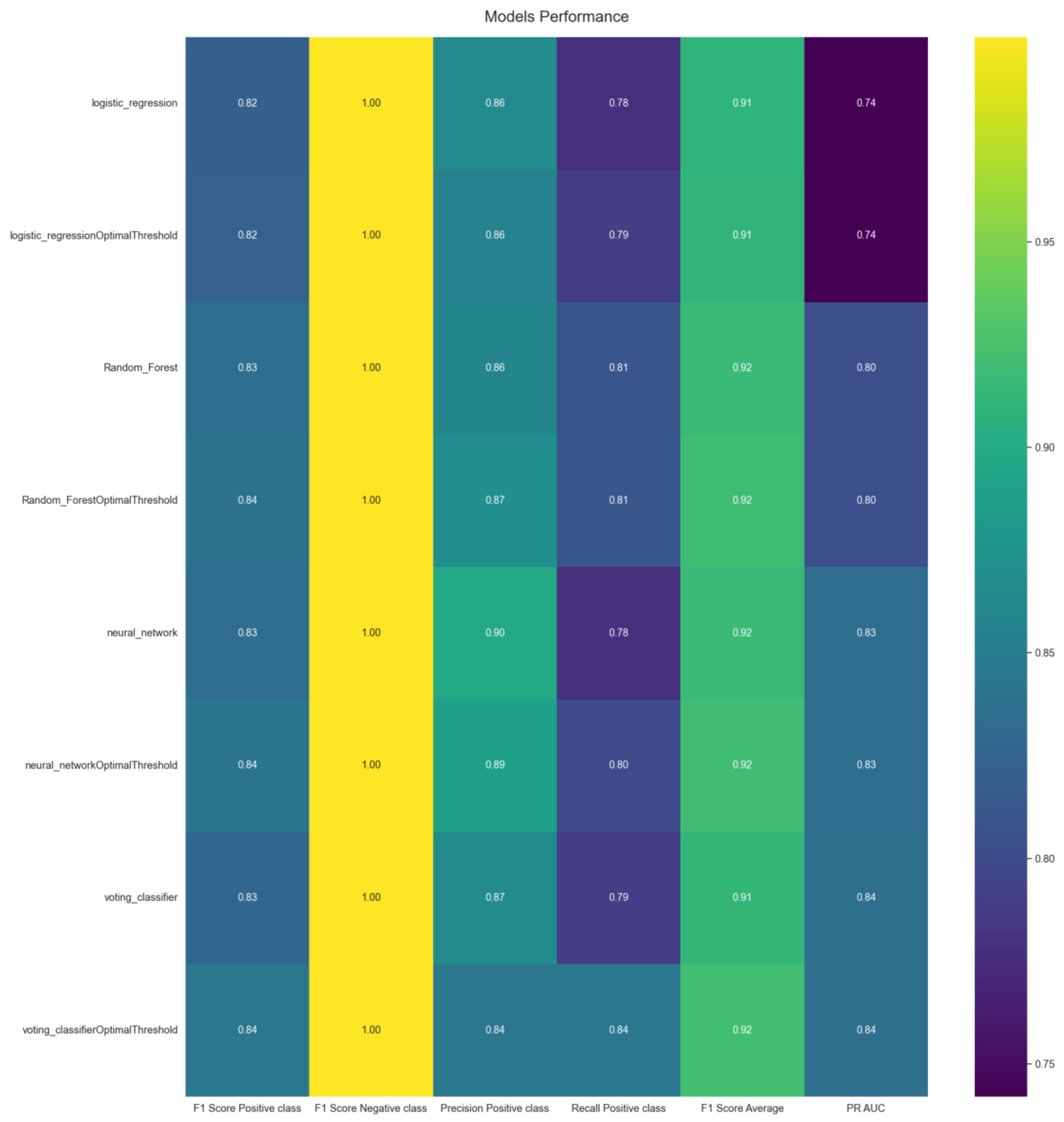
3. Multi-Layer Perceptrons (MLP) - Neural Networks

The MLP achieved the best balance between training and validation performance.

The Stochastic Gradient Descent (SGD) optimizer outperformed the Adam optimizer.

The Sigmoid activation function delivered better results than ReLU.

This combination of models allowed the voting classifier to leverage the strengths of each approach, yielding robust performance across the imbalanced dataset.



2.1 Using Focal Loss

Focal Loss is a loss function specifically designed to handle class imbalance, particularly in tasks like object detection. Introduced in the paper "Focal Loss for Dense Object Detection," it

$$\mathrm{FL}(p_t) = -\alpha_t (1 - p_t)^{\gamma} \log(p_t).$$

emphasizes difficult-to-classify examples while reducing the impact of easily classified ones. This is achieved using two parameters, α (alpha) and γ (gamma).

Parameter Tuning:

Experimented with various α values (0.80–0.99, step 0.05) and γ values (0–4, step 1).

Training Stability:

Higher values of γ (e.g., 5–7) led to noisy loss curves and unstable training.

Training stability improved with Batch Normalization and switching from SGD to Adamw optimizer.

Trade-off Between Precision and Recall:

Higher α values resulted in better recall for the positive class but reduced precision.

Conversely, lower α values improved precision but reduced recall.

Note : SMOTE and Under Sampling tend TO do less performance to all models