Pattern Recognition and Machine Learning
Minor Project
"Credit Card Fraud Detection"

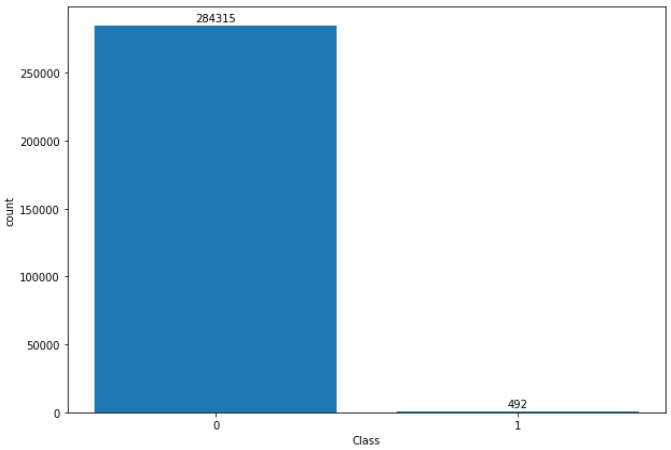
Team Members:-
Ashudeep Dubey (B21EE090)
Tanish Pagaria (B21AI040)
Vinay Vaishnav (B21EE084)

Problem Statement

Credit risk is associated with the possibility of a client failing to meet contractual obligations, such as mortgages, credit card debts, and other types of loans. 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. These datasets are hard to handle. You have to predict whether, given the details about the credit card, whether it is real or fake.

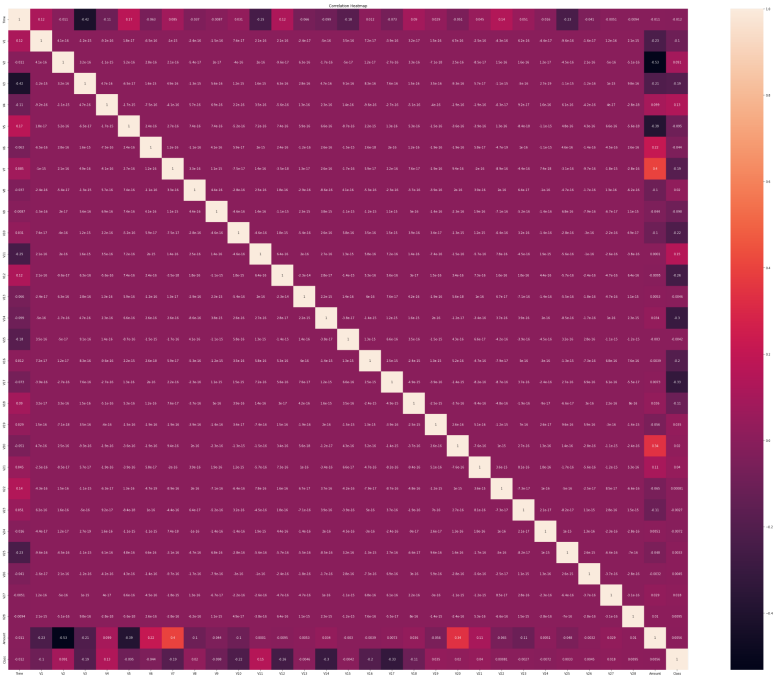
Data Analysis, Preprocessing & Visualization

The dataset did not contain any NULL entries or NaN values. The dataset was heavily skewed. The positive class data entries were only 492 as compared to 284315 negative class entries.



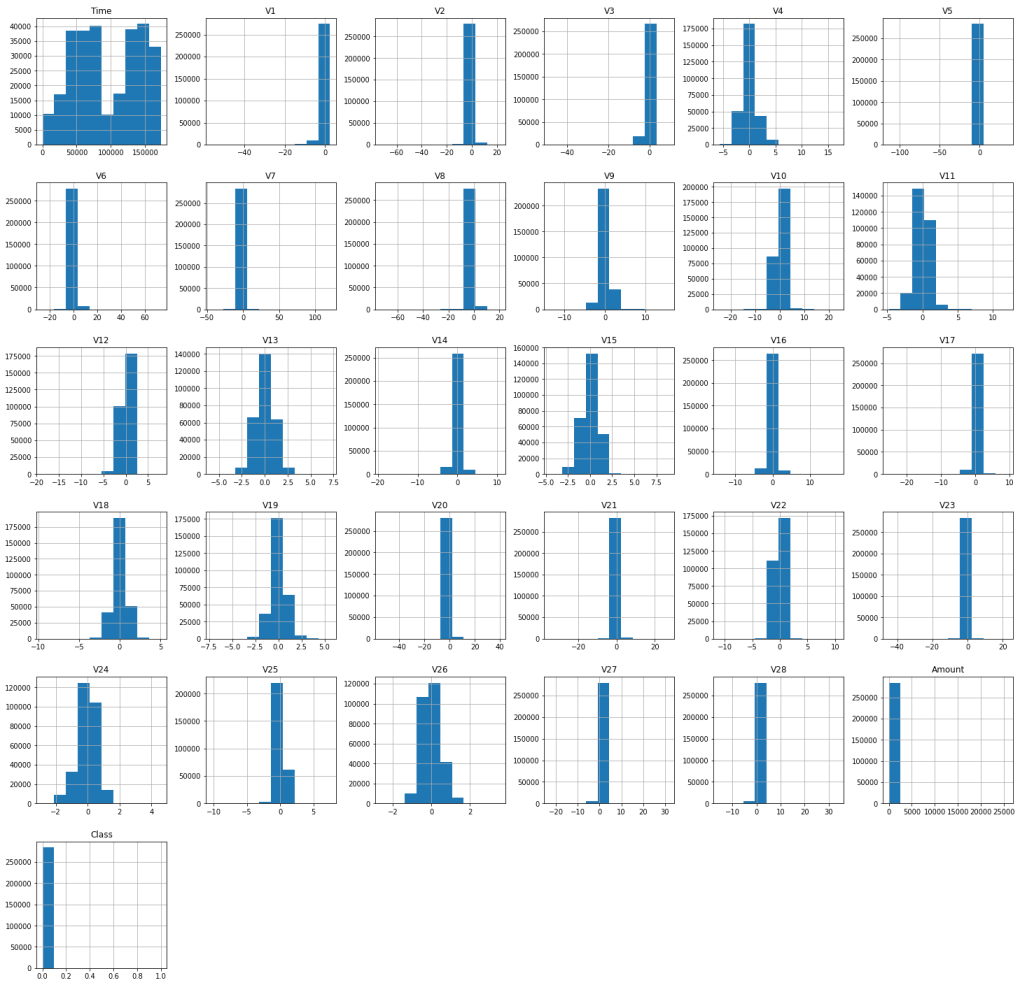
Bar plot showing class count comparison

We plotted the correlation heatmap among the features in order to identify if any features showed strong correlation.



Correlation heatmap

However, none of the features showed any significant correlation with each other.



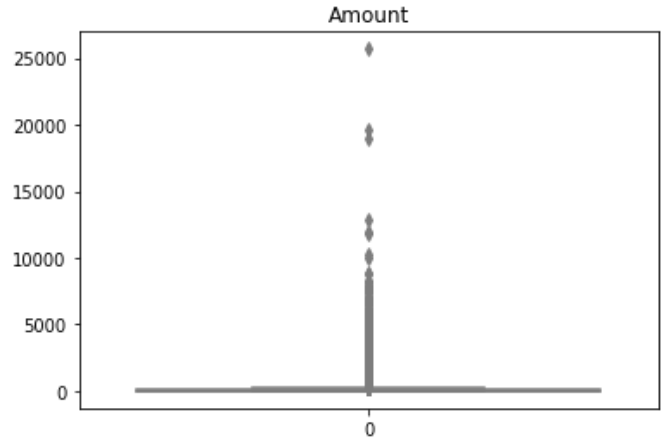
Histogram plot of the dataframe

The ***"Time"*** column, which did not show any significant utility in the classification task, was dropped.

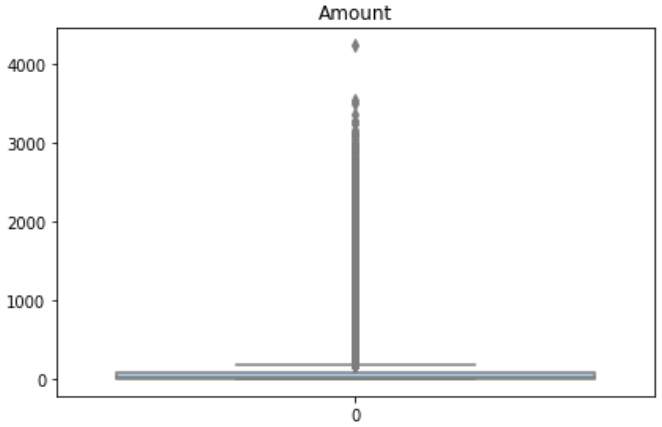
The ***"Amount"*** column of the dataset was standardized so that all the features were in a similar range. Scaling the data helps with better pattern recognition in the data and reduces the impact of outliers (if any).
The dataset was split into training and testing sets in an 80:20 ratio.

Outlier Detection

Isolation Forest technique was employed for detecting the outliers in the dataset. Isolation Forest can handle high-dimensional feature spaces, identify outliers without relying on assumptions about the underlying distribution of the data, and improve the performance of machine learning algorithms by creating a more representative dataset.
By varying the hyperparameters, 2849 anomalies were detected (which is not a very significant number considering the dataset). It was noted that none of these anomalies had any fraudulent data entries.



Before outlier removal



After outlier removal

Methods to handle imbalanced dataset

Under-sampling Technique

It is a technique in which we keep all the data points from the minority class and reduce the majority class points and try to balance the imbalanced data we have. Although, because of this, there is a high probability of losing important data, our model may not perform well.

Under-sampling method implementation (from scratch):
First, we found the number of minor class data points, and using that, we made a sample of the majority class and made final data by concatenating minor class points and samples. We saved these data points so that we could use them later to train models and compare the scores of different classification techniques.

Over-sampling Technique

It is a technique in which we keep all the data points of the majority class and increase the size of the minority class to make the dataset balanced.

Over-sampling method implementation (from scratch):
First, we found the number of major class data points, and using that, we made a sample of the minor class and made final data by concatenating the major class points and samples. We saved these data points so that we could use them later to train models and compare the scores of different classification techniques.

Classification techniques

The following classification techniques were employed on the three variations of the dataset (standardized, undersampled, and oversampled). We created customized functions from scratch in order to show the classification results of the techniques.
The primary evaluation metrics considered in the comparison included the F1-score, the accuracy scores, and the classification report.
F1-score was given a special preference in this case because accuracy alone was not an efficient criteria in determining the model's performance because of the highly imbalanced data.

Linear Discriminant Analysis

We decided to employ this technique because of the high dimensionality of the dataset. It is effective at reducing the dimensionality of the feature space while preserving the discriminatory power between fraudulent and legitimate transactions.

Random Forest Classifier

It can deal with high-dimensional feature spaces with complex nonlinear relationships between the input variables. It can also deal with datasets that aren't balanced and give estimates of how important each feature is.

Bagging Classifier
We decided to use this technique because it can improve the performance of the base classifier by reducing overfitting and improving the stability of the model.

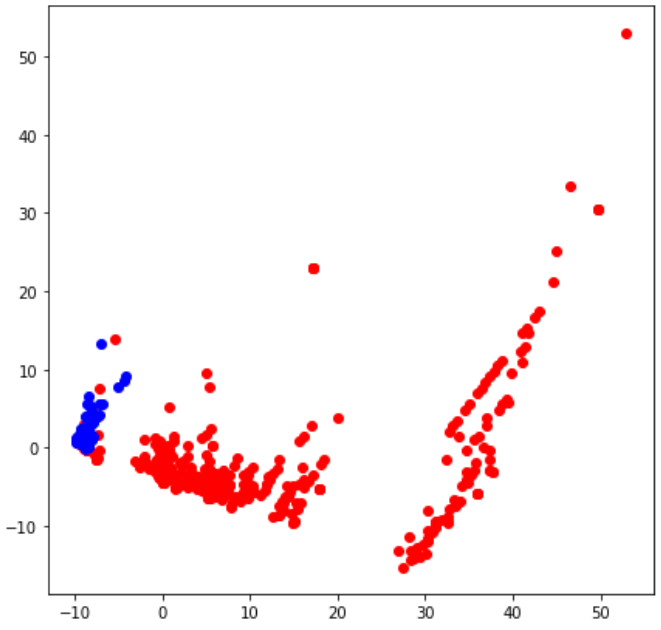
XGBoost Classifier
It iteratively trains weak learners based on the previous iterations to create a strong learner. It can handle missing data and imbalanced datasets and provide estimates of feature importance.

Manual ensemble methods (self-thought out)
Approach-1
It deals with splitting all the non-fraudulent cases into a certain number of batches and then adding all the fraudulent cases to each of the batches.

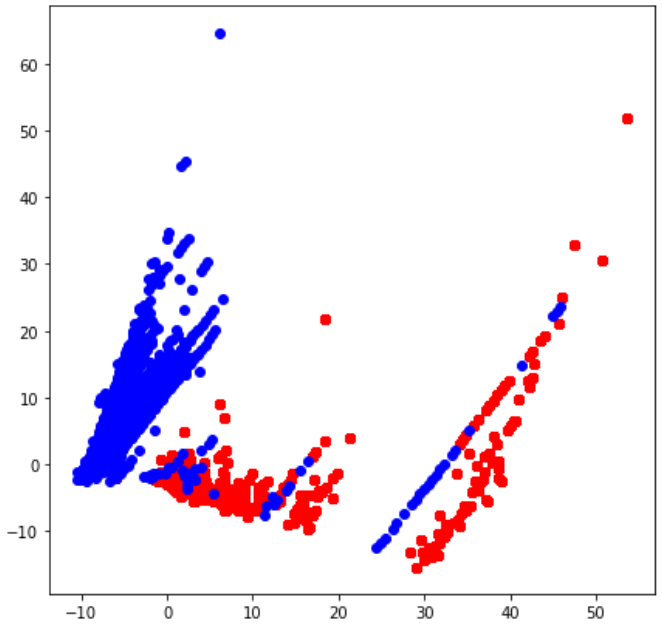
Approach-2
It deals with splitting all the non-fraudulent and fraudulent cases into a given number of batches (say "n") and randomly concatenating them to create "n" datasets (by combining one fraudulent and one non-fraudulent batch each time (without replacement)).

We decided to use these 2 approaches because each of them provides datasets with different levels of randomness. It is important to note that the whole dataset used for making the batches was obtained only from the training dataset.
For each approach, we trained a desired number of the same classifier on the "n" batches. After training the classifiers, we performed majority voting. It involved finding the most frequently occurring prediction for each data point in all the classifiers and using it as the final prediction for that data point.

Classification using KNN on the undersampled and oversampled data(A different approach)
*(after applying **PCA(n_components=2)**)*



Scatter-plot of the undersampled data
after applying PCA (n_components=2)



Scatter-plot of the oversampled data
after applying PCA (n_components=2)

KNN on undersampled data with PCA applied
On plotting the scatter plot on undersampled data with PCA applied, we could see that the classes formed two different clusters. And hence, we thought of applying KNN to this data. The f1-score obtained was: 0.030690537084398978

KNN on oversampled data with PCA applied
On plotting the scatter plot, we could see two different clusters again. This time the clusters were of comparable sizes for both classes. The f1-score obtained in this case was: 0.7553191489361704

It was observed that the oversampled data gave a better f1-score and hence a better performance. This was just a new approach to the problem statement, even though the results were not up to the mark as compared to other techniques. This was done solely based on observations of the transformed data plots.

F1-scores for the various classification techniques used

Classifier	Original Standardized Data	Under-sampled Data	Over-sampled Data
Linear Discriminant Analysis	0.8021978021978022	0.18859649122807015	0.3003663003663004
RandomForest Classifier	0.8636363636363635	0.09464378575143007	0.8636363636363635
Bagging Classifier	0.8457142857142858	0.0764276513525118	0.8279569892473119
XGBoost Classifier	0.8681318681318683	0.06947697111631537	0.47368421052631576
Ensemble Method - 1	0.8205128205128205	0.025453560790685083	0.8449197860962567
Ensemble Method - 2	0.7810650887573964	0.16697588126159554	0.8645833333333334

Final pipeline creation

In order to create the final pipeline for the problem statement, based on our testing, we decided to employ the XGBoost classifier for the classification task. A model was trained on the complete dataset after standardizing the "Amount" column and dropping the "Time" column. A function was defined that would return the prediction of the model after preprocessing the input dataset.
The preprocessing would involve dropping the "Time" column and standardizing the "Amount" column (as done with the training dataset). The predictions of the model would be the final result.