PRML
Minor Project Report File

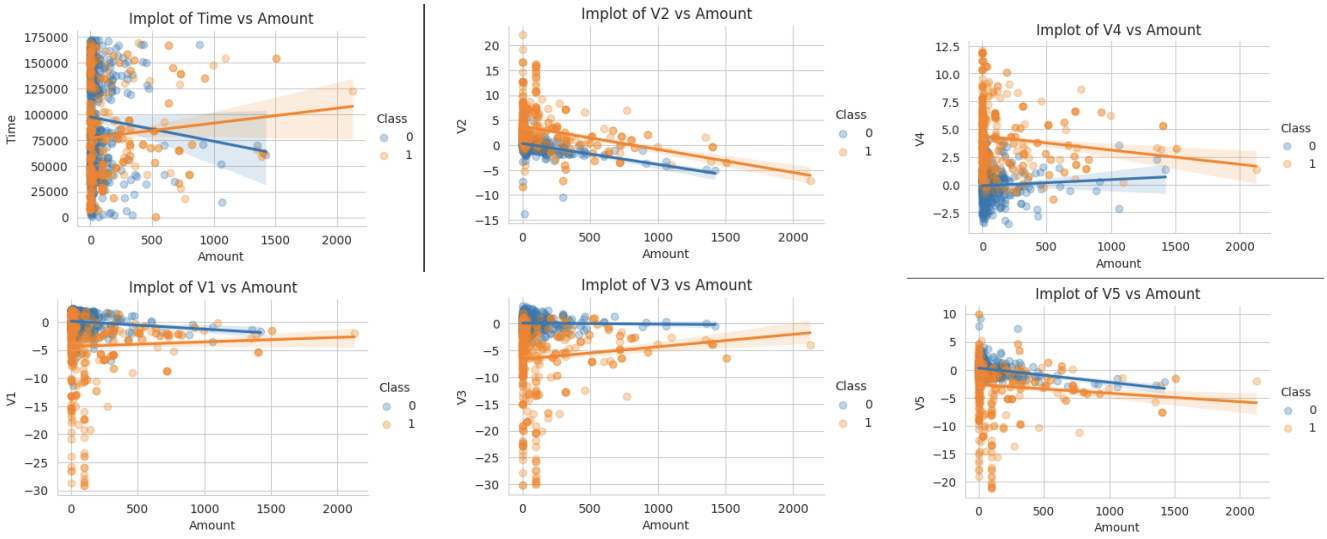
Jatin Lohar (B21CS091)
Himanshu Gaurav (B21EE025)
Himesh Dhaka (B21EE026)

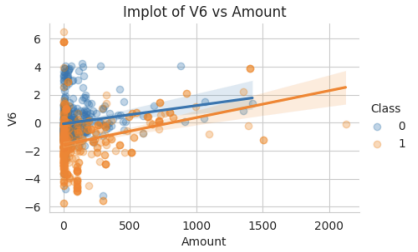
Importing the necessary libraries

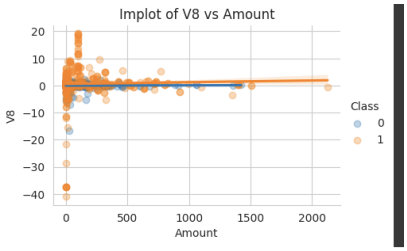
We imported numpy for numerical computing, pandas for data manipulation, seaborn for data visualization, and plotly.express for interactive visualization.

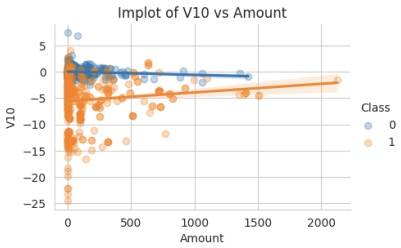
Loading data from the Google Drive

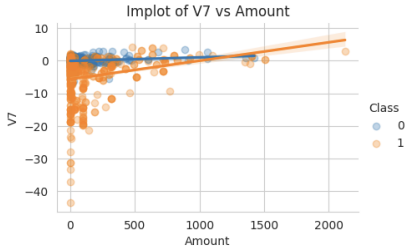
Mounting Google Drive on Collab and then reading creditcard.csv into pandas,we show the first 5 rows of the DataFrame to check data loading.

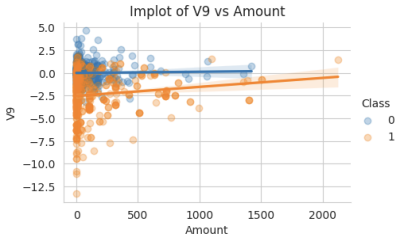


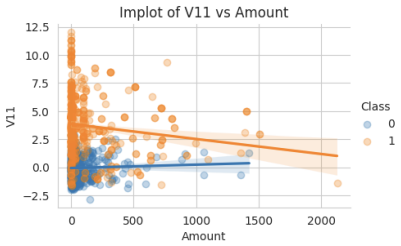

Implot of V6 vs Amount
V6
Amount
Class
0
1

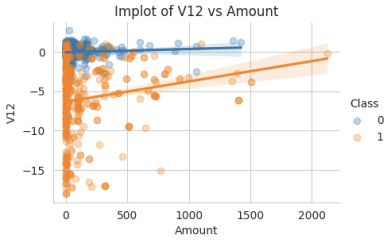

Implot of V8 vs Amount
V8
Amount
Class
0
1

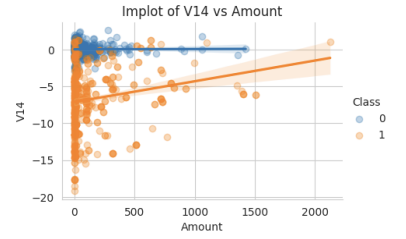

Implot of V10 vs Amount
V10
Amount
Class
0
1

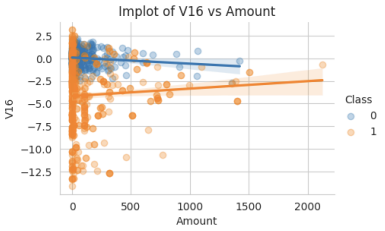

Implot of V7 vs Amount
V7
Amount
Class
0
1

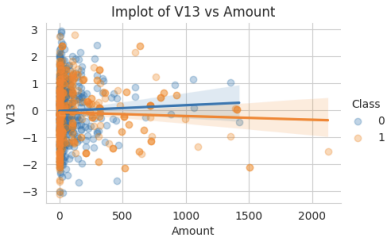

Implot of V9 vs Amount
V9
Amount
Class
0
1

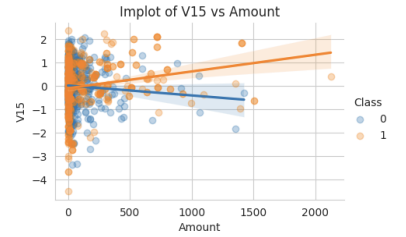

Implot of V11 vs Amount
V11
Amount
Class
0
1

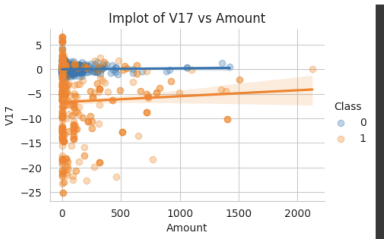

Implot of V12 vs Amount
V12
Amount
Class
0
1

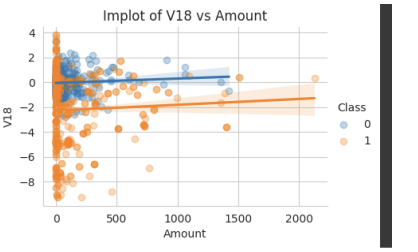

Implot of V14 vs Amount
V14
Amount
Class
0
1

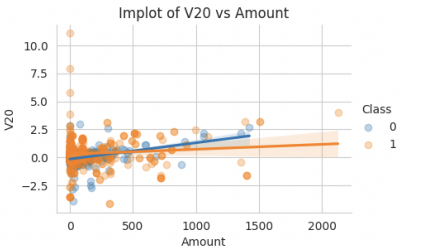

Implot of V16 vs Amount
V16
Amount
Class
0
1

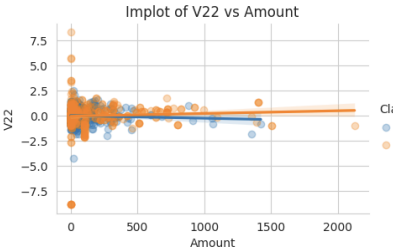

Implot of V13 vs Amount
V13
Amount
Class
0
1

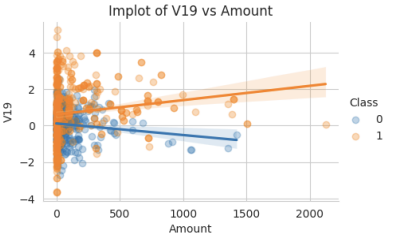

Implot of V15 vs Amount
V15
Amount
Class
0
1

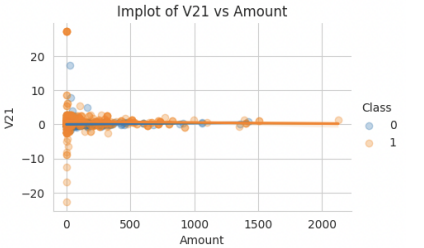

Implot of V17 vs Amount
V17
Amount
Class
0
1

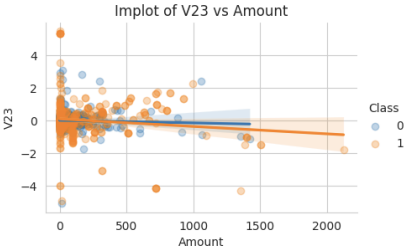

Implot of V18 vs Amount
V18
Amount
Class
0
1

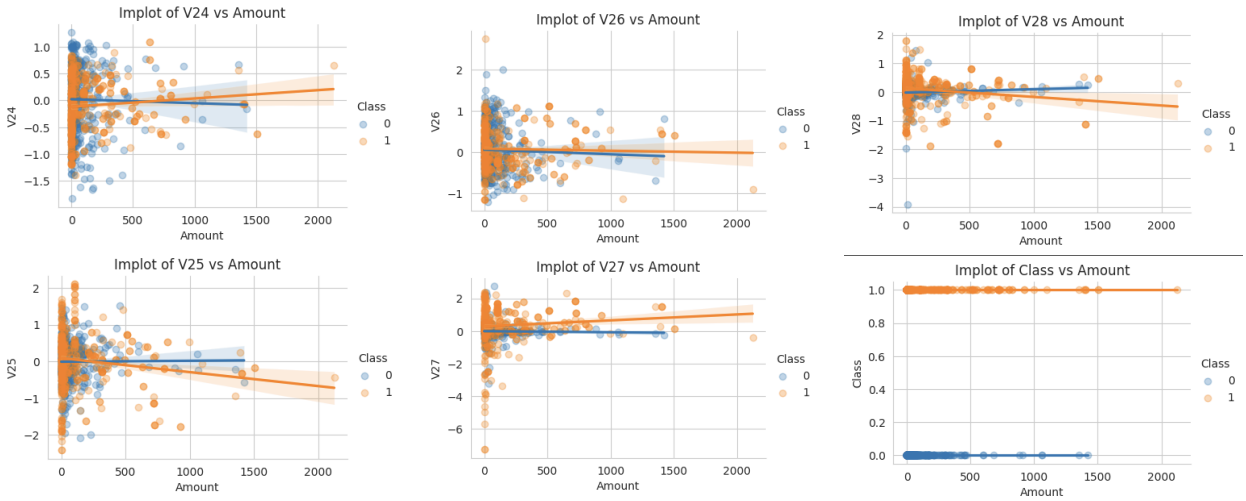

Implot of V20 vs Amount
V20
Amount
Class
0
1

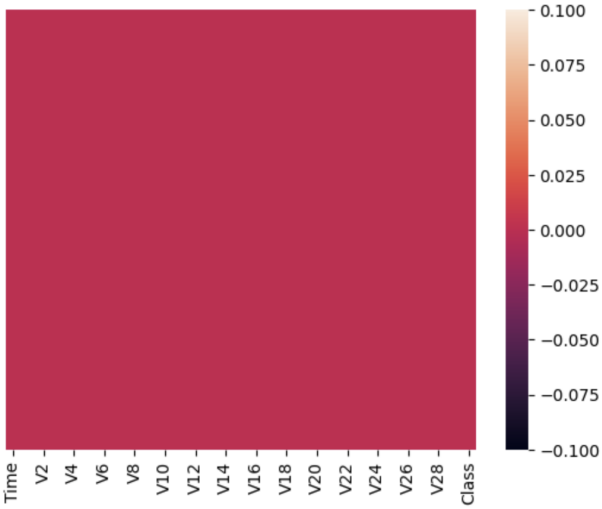

Implot of V22 vs Amount
V22
Amount
Class
0
1

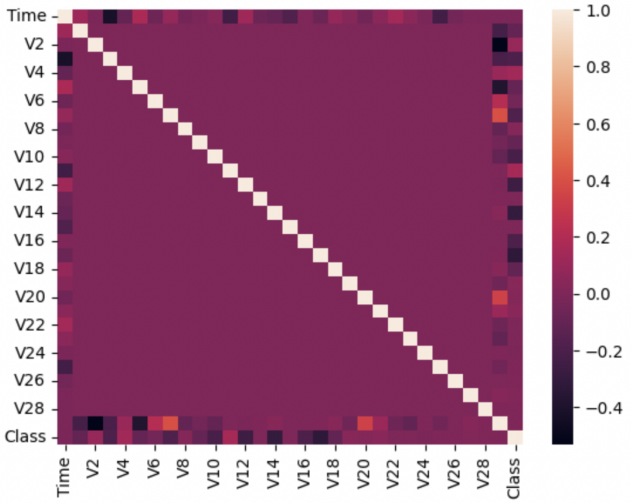

Implot of V19 vs Amount
V19
Amount
Class
0
1

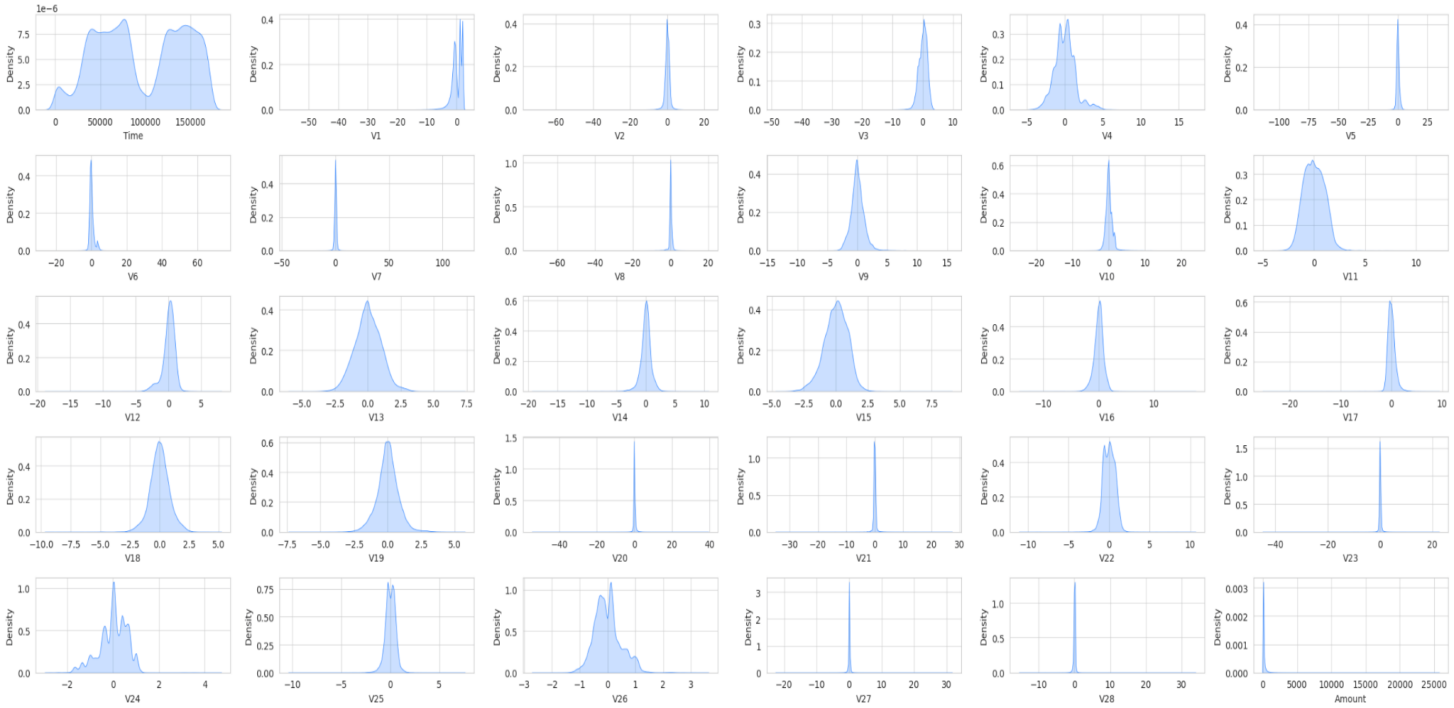

Implot of V21 vs Amount
V21
Amount
Class
0
1


Implot of V23 vs Amount
V23
Amount
Class
0
1


Implot of V24 vs Amount
Implot of V26 vs Amount
Implot of V28 vs Amount
Implot of V25 vs Amount
Implot of V27 vs Amount
Implot of Class vs Amount
Amount
Class
0
1


Time
V2
V4
V6
V8
V10
V12
V14
V16
V18
V20
V22
V24
V26
V28
Class


Time
V2
V4
V6
V8
V10
V12
V14
V16
V18
V20
V22
V24
V26
V28
Class



Random under-sampling to create a balanced dataset

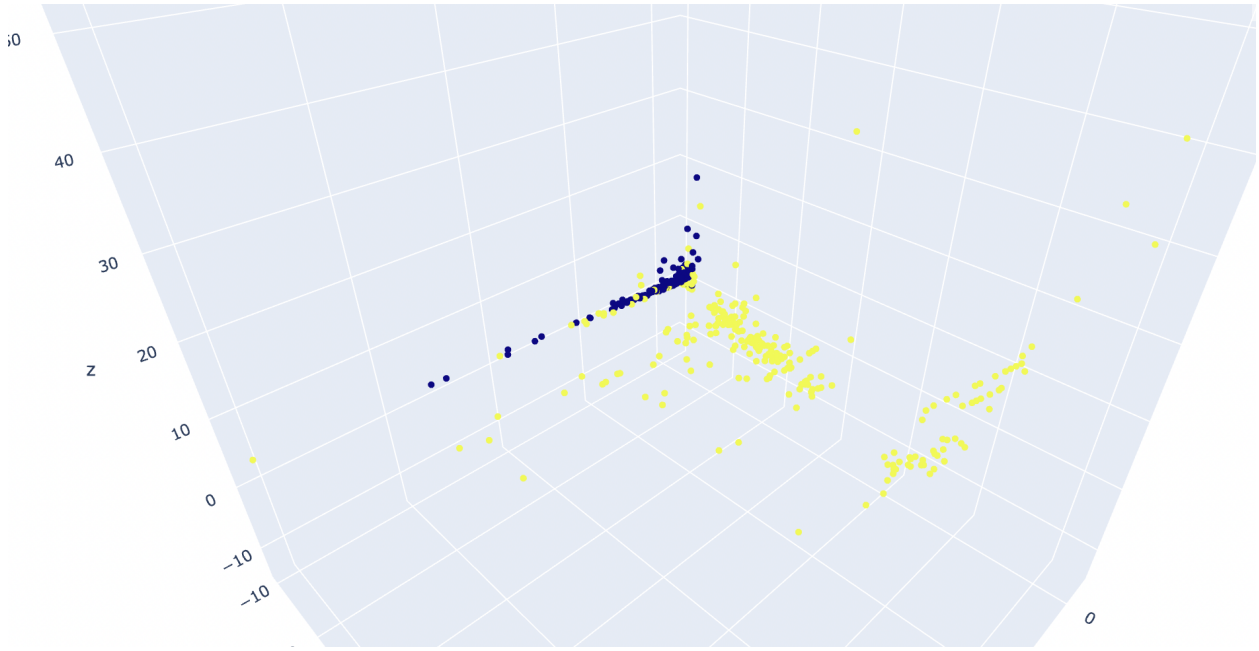
We under-samples the DataFrame to balance fraud and regular transactions for fraud detection machine learning models.Firstly we count fraud transactions in the data DataFrame and put the result in number records fraud.We next pick all fraud transaction indices and put them in a NumPy array called fraud indices. We keep all regular transaction indices in a NumPy array called normal indices.We randomly select a subset of normal transaction indices and randomly pick a subset of fraud transaction indices with replacement that has the same length as the fraud indices array and then concatenate the random fraud indices and random normal indices arrays into a NumPy array called under sample indices, which reflects the DataFrame's under-sampled indices.This shows the under sample indices array, which should be 1D and double the length of number records fraud.Thereby we construct a test dataset using a portion of the under-sampled data to assess the performance of a machine learning model trained on the remaining data.

Dimensionality reduction using PCA

PCA reduces the feature space to 3 dimensions for the test dataset X test. Plotly builds a 3D scatter plot of modified data.

We created a 3-component PCA class object to convert the feature matrix X test into a lower-dimensional space.This turns the dataset into a 3-dimensional feature matrix X1 creating a new Plotly figure object with a 3D scatter plot trace. The 3-dimensional feature matrix X1

provides the x, y, and z coordinates, and the Y test vector labels define the point colors.We reduced dimensionality using PCA and visualized the modified data in a 3D scatter plot to find clustering patterns and class separation (fraudulent vs. non-fraudulent transactions).



Implementation of LDA from Scratch

We have created an LDA class for dimensionality reduction using LDA.This class allows LDA-based dimensionality reduction with customizable explained variance and dimensions. It can be instantiated, fitted, and used to change fresh data. The scatter within and scatter between functions directly access the within-class and between-class scatter matrices.

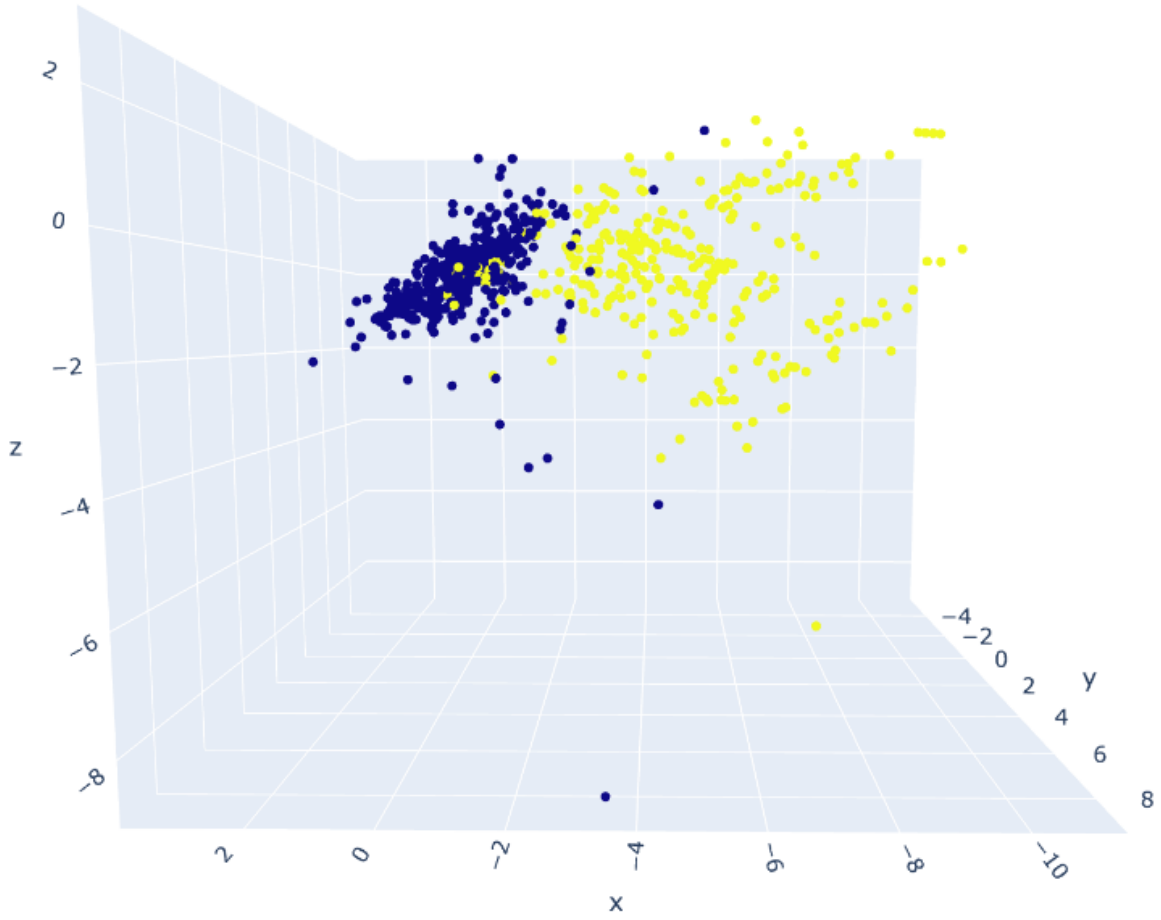
Class methods are :

__init (self): initializes an empty list named idx.
fit(self, X, Y, var=0.95): X is the input feature matrix, Y is the output target vector, and var is the required percentage of explained variance (default is 0.95). It calculates the class means and grand mean for each class in Y. These methods compute the within-class and between-class scatter matrices. Lastly, it calculates the ratio of these matrices' eigenvectors and eigenvalues and arranges them in decreasing order.

self.transform(X, n): This approach requires X (the input feature matrix) and n (the number of dimensions to reduce to). It multiplies the modified feature matrix X by the projection matrix W using the n eigenvectors with the greatest eigenvalues.
scatter within(self): Returns the within-class scatter matrix.
scatter between(self): Returns the between-class scatter matrix.

Visualizing the distribution in the LDA transformed space

Plotly creates a 3D scatter plot using this code. It depicts the data points in the modified feature space following LDA on the under-sampled dataset. The LDA transform function on the test data produces the X new variable. Class labels are in the Y test variable. Color-coding shows data point class designations in the graphic. Red denotes fraud and blue represents non-fraud. The plot shows the data points' distribution in the converted feature space and helps us comprehend the LDA transformed space's class separation.



Implementation of bagging ensemble method from Scratch

The class BaggingClassifier provides a bagging ensemble classification algorithm. Bagging includes producing several random subsamples of the original dataset with replacement and training a base model on each subsample. Bagging reduces overfitting to increase model stability and accuracy.

The BaggingClassifier class's function accepts two arguments: clf, the base classifier type (default is DecisionTreeClassifier()), and n estimators, the number of base classifiers to utilize (default is 30).
The fit technique generates n estimators balanced undersampled subsamples of the training data to train the ensemble of base classifiers. The ensemble adds a basic classifier of type clf trained on each subsample.
The predict_proba technique predicts class probabilities for each sample in a matrix X of feature vectors. Applying each ensemble base classifier to the input feature vector and averaging the projected probabilities yields the prediction.

Creating an ensemble model consisting of multiple base classifiers

A Decision Tree Classifier, a K-Nearest Neighbor Classifier, and a Random Forest Classifier

The class model comprises DecisionTreeClassifier(), KNN(n neighbors=5), and RandomForestClassifier(n estimators=10, max depth=5) classification models. It also lists itself. Bag models for Bagging models. The fit function trains classification and Bagging models using training data. The predict function predicts input data class labels using learned Bagging models.
The BaggingClassifier class with the classification model and number of estimators creates the Bagging models (default is 10). The fit function randomly under-samples the majority class (0) to balance class distribution and trains the given classification model on the under-sampled data for each Bagging model.
For each input data point, Bagging models predict the class probability using predict_proba, and the class with the greatest sum of probabilities is the anticipated class label.

LDA transforms data for multiple models

At first we construct an empty model list then an LDA model is fitted to X and Y using fit().
The LDA transformation fits a model instance on modified data and adds it to the list of models. The predict() function uses LDA for various feature dimensions to predict class labels for incoming data. Summating and thresholding the predictions yields the final forecast. If the amount is under 9, the forecast is 0, otherwise 1.

Printing the f1 score, accuracy score, and confusion matrix

Based on main model instance md predictions and Y test true labels, we produce the F1 score, accuracy score, and confusion matrix.