RELIABLE CREDIT CARD CLIENTS

Student Name: Yash Mittal

Course & Batch: Artificial Intelligence and Data Science B1

Enrollment Number: 02619011921

Email ID: yashmittal052@gmail.com

Contact number: 7011590709

Code:

https://colab.research.google.com/drive/1Pbe-8cis2uRN-o667zTJh-1zydJJKs1z?usp=sharing

Data Set:

https://archive.ics.uci.edu/dataset/350/default+of+credit+card+clients

Website Link:

https://y-a-s-h-m-i-t-t-a-l.github.io/rccc/index.html

Abstract

The project intends to showcase comparison between 10 different classifiers on our chosen dataset, including the ones with ensemble learning techniques. The dataset employed in this study represents the behavior of credit card holders in China during the year 2015. It encompasses essential demographic information of the cardholders as well as their billed transactions over a six-month period. The dataset was preprocessed, and scaled using standardization. Further, we split the same data into training (75%) and testing (25%) to train our model and test along. The final task to accomplish was classification of ‘default payment next month’ as Yes(1) or No(0). It, in general, refers to the prediction whether a person will make the outstanding default payment of their balance amount next month or not.

We evaluate and compare multiple classifiers, and shed light on their effectiveness in tackling the credit card default prediction problem. Ensemble learning methods, known for their ability to combine multiple models to improve overall performance, are added specially to assess their potential advantages in this specific context. We use accuracy score and Area under curve as the metrics to compare the results, and find that Support Vector Machine (SVM) is a good classifier to predict the results in the case when such data of 6 months is given.

The findings of this study can potentially contribute to the development of more reliable and robust credit scoring systems, which are crucial for the financial industry to effectively manage credit risks. Ultimately, the project seeks to optimize the selection of credit card customers, ensuring a more stable and profitable customer base for credit card issuers in the long term.

Keywords

- Classifiers: Algorithms or models that can be trained to assign labels or categories to data based on certain patterns or features. They are used to classify data into different classes or groups.

- Ensemble learning techniques: Methods that involve combining multiple individual models or classifiers to improve prediction accuracy or robustness.

- Dataset: A collection of structured data that represents a set of observations or measurements. In this passage, the dataset contains information about credit card holders in China, including demographic data and transaction information.

- Standardization: The process of scaling or transforming data to have a consistent range or distribution, often done to facilitate accurate comparisons or modeling.

- Preprocessed: Refers to the steps taken to clean, transform, or manipulate the data before it is used for analysis or modeling.

Introduction

The effective prediction of credit card default payments plays a vital role in the financial industry's ability to manage credit risks and maintain stable and profitable customer bases. In this study, we aim to showcase the comparison between 10 different classifiers, including those utilizing ensemble learning techniques, on a carefully chosen dataset representing the behavior of credit card holders in China during the year 2015. We include Ensemble learning models to see if their strategic abilities show us drastic improvements. Foremost, there would be study on the dataset, we would make it ready and fit for our models, predict and calculate results with visualizing graphs.

Dataset

The dataset used in this research is taken from https://archive.ics.uci.edu/dataset/350/default+of+credit+card+clients, uploaded on GoogleDrive and read from there itself. It focuses on analyzing the credit card default payment behavior of individuals. The response variable, default payment, is represented as in classes, with "Yes" denoted as 1 and "No" as 0. To do the needed study, 23 factors in columns are incorporated along with more than 30,000 samples. The dataset is already label encoded as follows -

X1: Amount of the given credit (NT dollar): This variable includes both the individual consumer's credit and any supplementary credit associated with their family.

X2: Gender: It is a categorical variable with two options, where 1 represents male and 2 represents female.

X3: Education: This variable categorizes the education level of the individuals into four categories: 1 for graduate school, 2 for university, 3 for high school, and 4 for others.

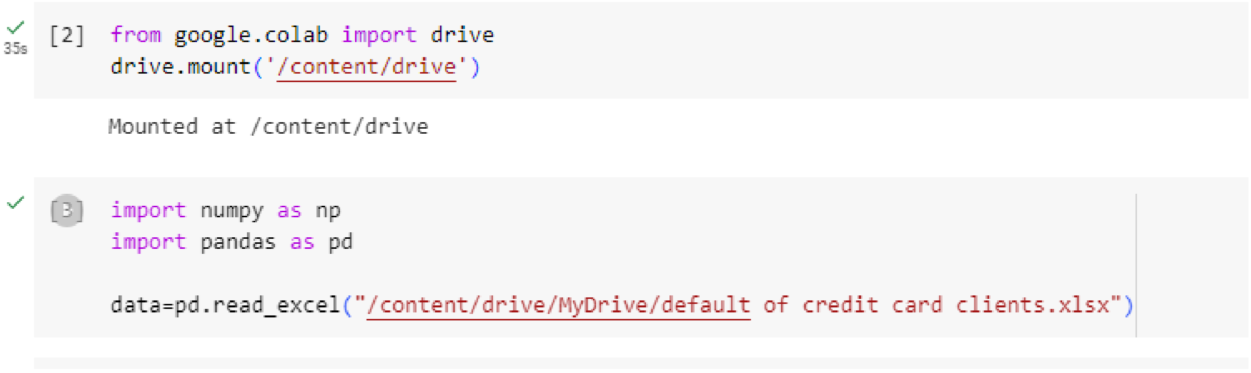
X4: Marital status: It categorizes the marital status of individuals into three categories: 1 for married, 2 for single, and 3 for others.

X5: Age: Represents the age of the individuals in years.

X6 - X11: History of past payment: These variables track the past monthly payment records from April to September 2005. Each variable represents the repayment status for a specific month. The measurement scale for the repayment status is as follows: -1 for paying duly, 1 for payment delay for one month, 2 for payment delay for two months, and so on up to 9 for payment delay for nine months and above.

X12 - X17: Amount of bill statement (NT dollar): These variables capture the amount of the bill statement for each corresponding month, from September 2005 (X12) to April 2005 (X17).

X18 - X23: Amount of previous payment (NT dollar): These variables indicate the amount paid by individuals in each corresponding month, from September 2005 (X18) to April 2005 (X23).



```
35s [2] from google.colab import drive
        drive.mount('/content/drive')

        Mounted at /content/drive
```

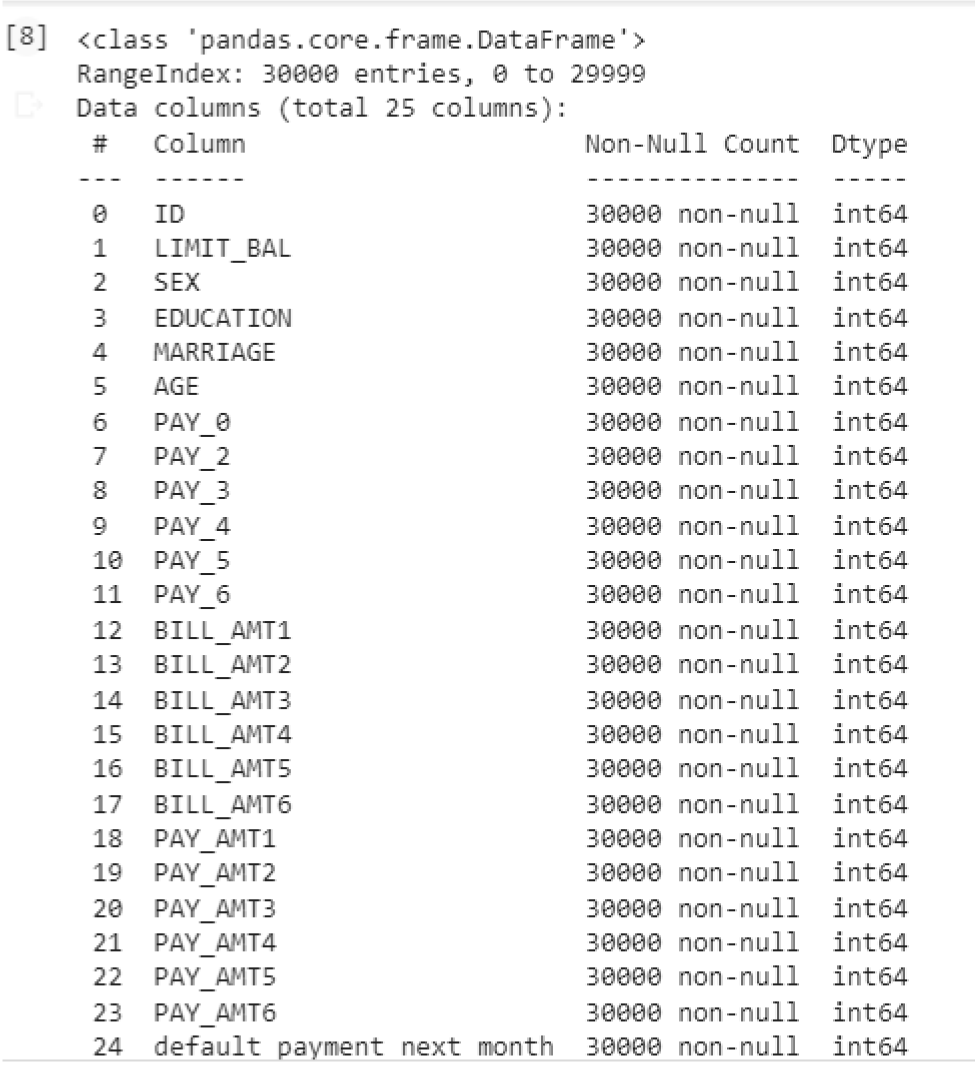
```
    [3] import numpy as np
        import pandas as pd

        data=pd.read_excel("/content/drive/MyDrive/default of credit card clients.xlsx")
```

Preprocessing

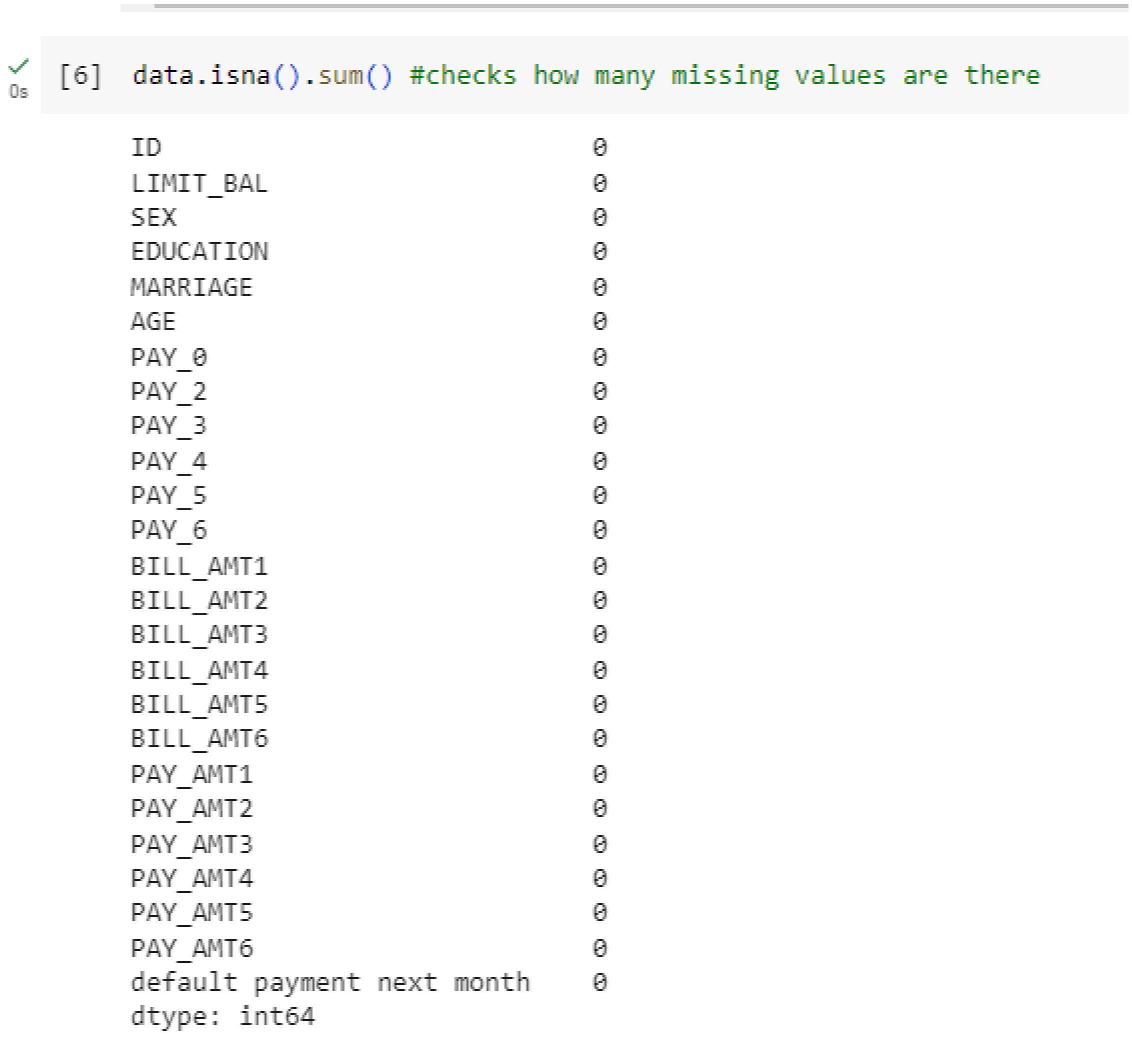
The dataset was analyzed thoroughly of its characteristics, so that we can make the necessary changes and make it suitable to train our ML model.

The first step we do is making a dataframe, taking a glimpse of its values using .head() operation, .describe() operation and eventually detailed information using .info() operation. It showcases that all our values in the various columns are of Dtype-int64 and we also checked the dataset that all the possible categorical values are already labeled encoded.



```
[8] <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 30000 entries, 0 to 29999
    Data columns (total 25 columns):
     #   Column                      Non-Null Count  Dtype
    ---  ------                      --------------  -----
     0   ID                          30000 non-null  int64
     1   LIMIT_BAL                   30000 non-null  int64
     2   SEX                         30000 non-null  int64
     3   EDUCATION                   30000 non-null  int64
     4   MARRIAGE                    30000 non-null  int64
     5   AGE                         30000 non-null  int64
     6   PAY_0                       30000 non-null  int64
     7   PAY_2                       30000 non-null  int64
     8   PAY_3                       30000 non-null  int64
     9   PAY_4                       30000 non-null  int64
     10  PAY_5                       30000 non-null  int64
     11  PAY_6                       30000 non-null  int64
     12  BILL_AMT1                   30000 non-null  int64
     13  BILL_AMT2                   30000 non-null  int64
     14  BILL_AMT3                   30000 non-null  int64
     15  BILL_AMT4                   30000 non-null  int64
     16  BILL_AMT5                   30000 non-null  int64
     17  BILL_AMT6                   30000 non-null  int64
     18  PAY_AMT1                    30000 non-null  int64
     19  PAY_AMT2                    30000 non-null  int64
     20  PAY_AMT3                    30000 non-null  int64
     21  PAY_AMT4                    30000 non-null  int64
     22  PAY_AMT5                    30000 non-null  int64
     23  PAY_AMT6                    30000 non-null  int64
     24  default payment next month  30000 non-null  int64
```

Next, we use the function .isna() to check any missing value. It returns us missing values as yes or no, so we take their sum to calculate the total no. of missing values in each column. Luckily, there were no missing values, else we would have to delete or impute the rows.

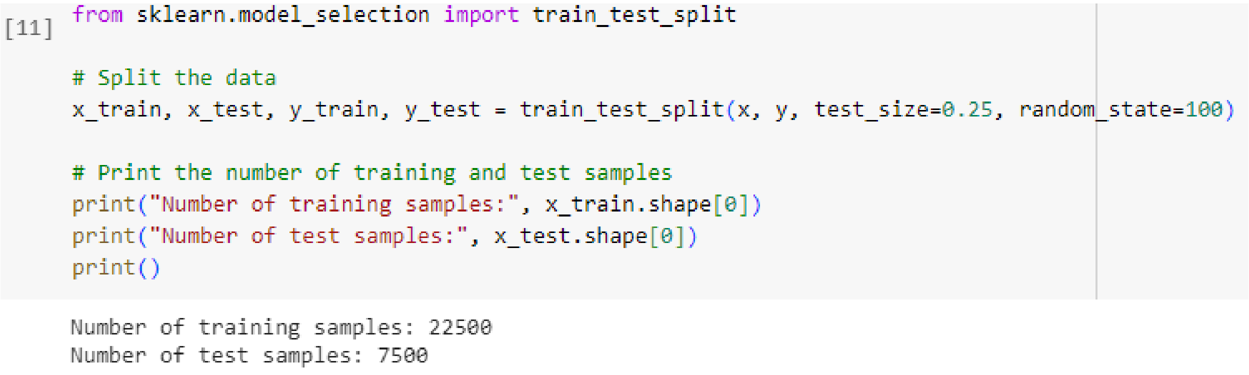


```
data.isna().sum() #checks how many missing values are there
```

```
ID                            0
LIMIT_BAL                     0
SEX                           0
EDUCATION                     0
MARRIAGE                      0
AGE                           0
PAY_0                         0
PAY_2                         0
PAY_3                         0
PAY_4                         0
PAY_5                         0
PAY_6                         0
BILL_AMT1                     0
BILL_AMT2                     0
BILL_AMT3                     0
BILL_AMT4                     0
BILL_AMT5                     0
BILL_AMT6                     0
PAY_AMT1                      0
PAY_AMT2                      0
PAY_AMT3                      0
PAY_AMT4                      0
PAY_AMT5                      0
PAY_AMT6                      0
default payment next month    0
dtype: int64
```

Then, we separate our attributes/features and the output test variable using the 'train_test_split' function of scikit-learn. The feature 'ID' just depicts the primary key of a sample and has negligible relation with the predicted output, so it is dropped.

Further, it is important that all the attributes share the same potential and parametric value, so we standardize the data using 'StandardScaler' of scikit-learn.



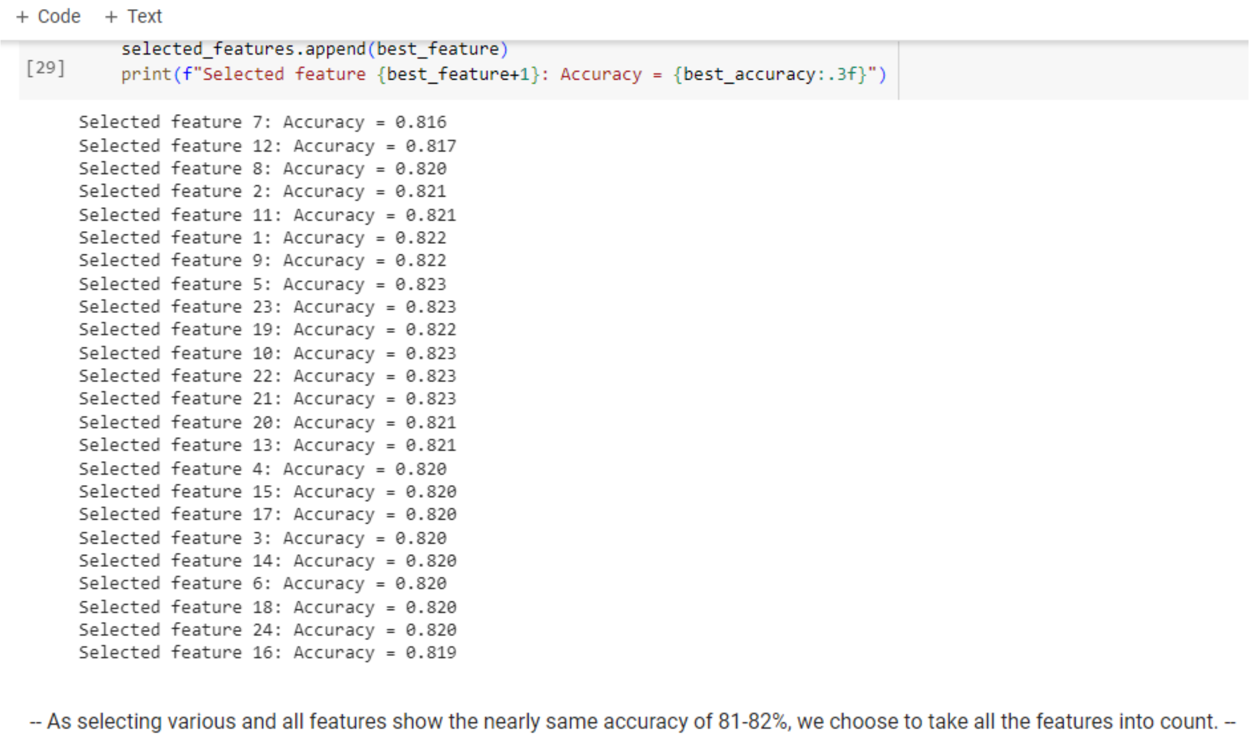
```
from sklearn.model_selection import train_test_split

# Split the data
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=100)

# Print the number of training and test samples
print("Number of training samples:", x_train.shape[0])
print("Number of test samples:", x_test.shape[0])
print()
```

```
Number of training samples: 22500
Number of test samples: 7500
```

Lastly, it is necessary to use the most relevant features to train and predict the model, so we did feature selection using the algorithm of Sequential forward selection and SVM classifier. All the features stood out and we could see an accuracy of 81-82%, so all the features are welcomed to give our training of the model.



```
    selected_features.append(best_feature)
    print(f"Selected feature {best_feature+1}: Accuracy = {best_accuracy:.3f}")
```

```
Selected feature 7: Accuracy = 0.816
Selected feature 12: Accuracy = 0.817
Selected feature 8: Accuracy = 0.820
Selected feature 2: Accuracy = 0.821
Selected feature 11: Accuracy = 0.821
Selected feature 1: Accuracy = 0.822
Selected feature 9: Accuracy = 0.822
Selected feature 5: Accuracy = 0.823
Selected feature 23: Accuracy = 0.823
Selected feature 19: Accuracy = 0.822
Selected feature 10: Accuracy = 0.823
Selected feature 22: Accuracy = 0.823
Selected feature 21: Accuracy = 0.823
Selected feature 20: Accuracy = 0.821
Selected feature 13: Accuracy = 0.821
Selected feature 4: Accuracy = 0.820
Selected feature 15: Accuracy = 0.820
Selected feature 17: Accuracy = 0.820
Selected feature 3: Accuracy = 0.820
Selected feature 14: Accuracy = 0.820
Selected feature 6: Accuracy = 0.820
Selected feature 18: Accuracy = 0.820
Selected feature 24: Accuracy = 0.820
Selected feature 16: Accuracy = 0.819
```

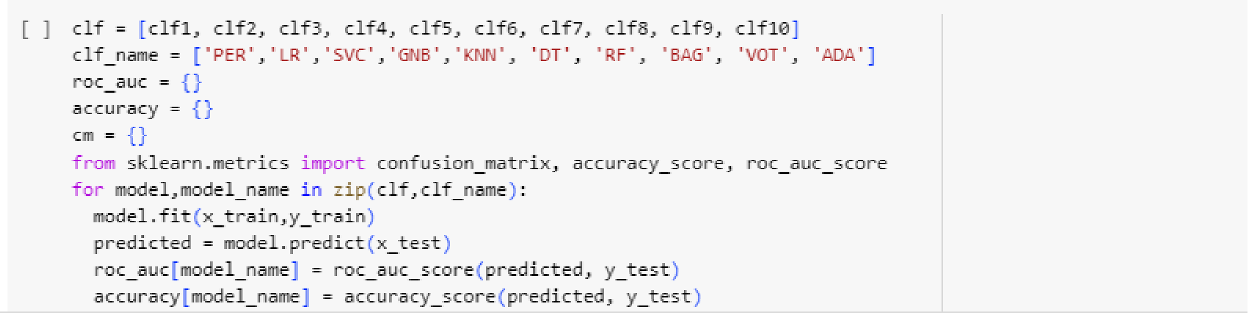
-- As selecting various and all features show the nearly same accuracy of 81-82%, we choose to take all the features into count. --

Model fitting

The data once preprocessed completely, is given to our machine learning models or classifiers. The classifiers we use and compare are:

1. Perceptron
2. Logistic Regression
3. Support Vector Machine
4. Gaussian Naive Bayes
5. KNeighborsClassifier
6. Decision Tree
7. Random Forest
8. Bagging
9. Voting (by Decision Tree, KNN and SVM)
10. AdaBoost

Imported them from the scikit-learn library, and labeled them as clf1-10 in the same order. We train and test each model one-by-one, and calculate the time, confusion matrix. accuracy, and area under the curve, for their result estimation. To accomplish this, we simply train our model with training data, and then predict the values of testing data. The time taken to do this task is noted, and then the predicted values are matched with the testing target to measure accuracy and auc_score.

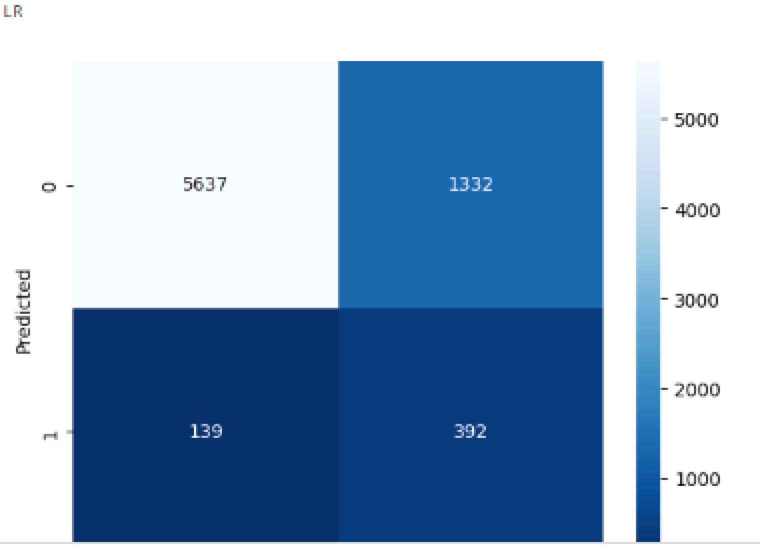


```
[ ]  clf = [clf1, clf2, clf3, clf4, clf5, clf6, clf7, clf8, clf9, clf10]
     clf_name = ['PER','LR','SVC','GNB','KNN', 'DT', 'RF', 'BAG', 'VOT', 'ADA']
     roc_auc = {}
     accuracy = {}
     cm = {}
     from sklearn.metrics import confusion_matrix, accuracy_score, roc_auc_score
     for model,model_name in zip(clf,clf_name):
       model.fit(x_train,y_train)
       predicted = model.predict(x_test)
       roc_auc[model_name] = roc_auc_score(predicted, y_test)
       accuracy[model_name] = accuracy_score(predicted, y_test)
```

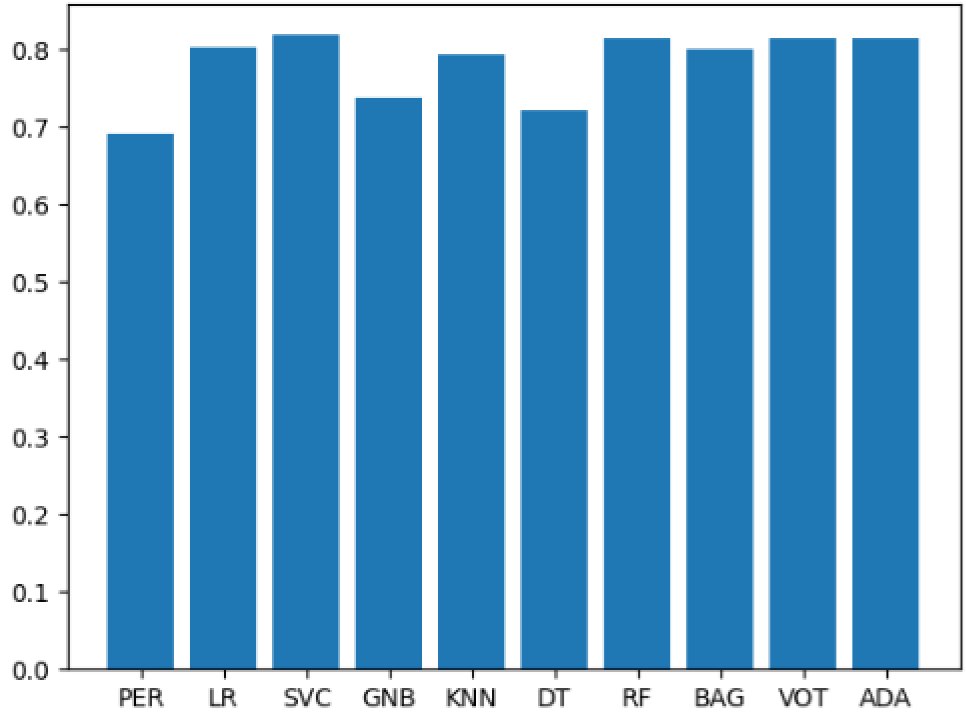
Results and Discussion

- We calculate and plot the Confusion Matrix, Accuracy Score and ROC Area under Curve for each classifier. We import libraries seaborn and matplotlib, to get a clear visual comparison of our results. We present the confusion matrix in a heatmap, and the other scores through a bar-graph.

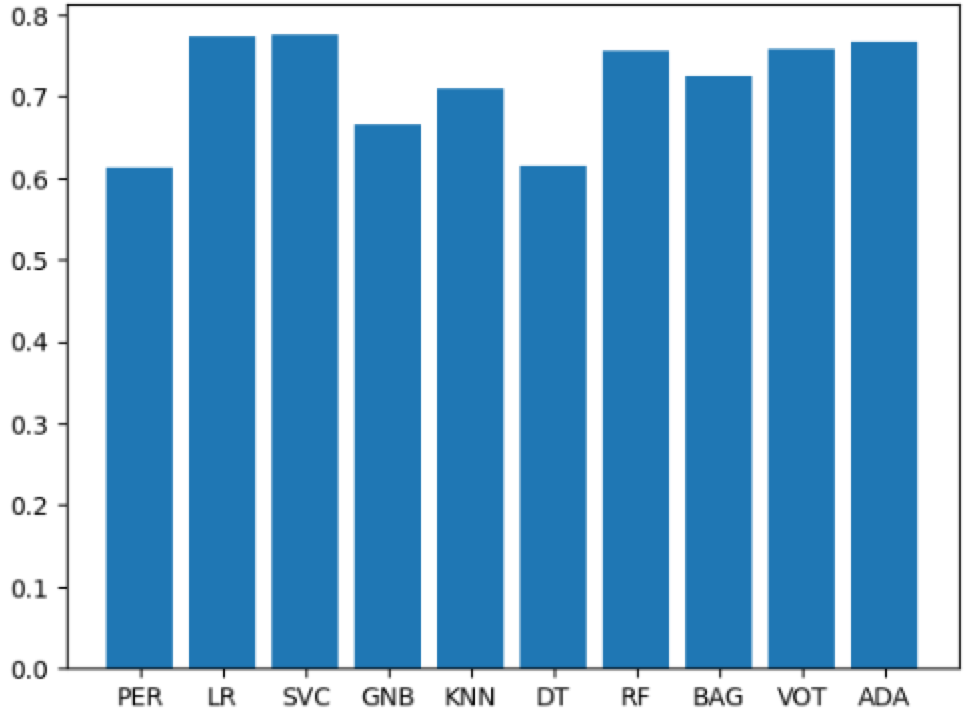
- From the confusion matrix, we observe that there is low accuracy in predicting the class'1' which states 'Yes', and there are more incorrect classifications of the class1 than the correct ones, for most of the classifiers.



- The accuracy scores are printed and plotted against a bar graph. Most of our classifiers stood at an accuracy of near about 80%, excluding Perceptron, Decision Tree and Gaussian Naive Bayes which showed an accuracy of even less than 75%.
  The worst accuracy is seen with Perceptron(69.17333333333333%) and the best with SVM(81.90666666666667%)



- The ROC AUC scores are printed and plotted against a bar graph. Most of our classifiers stood with an area of near about 75%, excluding Perceptron, Decision Tree and Gaussian Naive Bayes which showed an area of even less than 67%.
  Ideally the area should be 100% or 1. In this case, the least area is seen with Perceptron(0.6119525199068266) and the most with SVM(0.7752098520929918).



Conclusion and Future Scope

- The ensemble learning techniques, although known for their ability to combine multiple models to improve overall performance, performed relatively the same and couldn’t stand out any big in our prediction project.
- SVM showed us the maximum accuracy and area under the curve, so it concludes to be the best in predicting default payment when data of 6 months and similar features is given.
- The organizations can use this to accurately identify customers who are likely to default on their payments, and these organizations can take proactive measures to mitigate risks, such as adjusting credit limits, offering financial counseling, or implementing stricter approval criteria.
- The model has many false negative instants, which refers to the persons who would default payment but predicted they wouldn't, so the company may lose some potential clients but the false positive rate is much less to be taken at risk and the company would not suffer big losses.
- Further, there can be attempts with other classifiers and algorithms, or maybe adding some more crucial features to the dataset in order to make clearer and accurate results.

References

- UCI Machine learning repository https://archive.ics.uci.edu/dataset/350/default+of+credit+card+clients
- Yeh, I. C., & Lien, C. H. (2009). The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. Expert Systems with Applications, 36(2), 2473-2480.
- Baesens, B., Roesch, D., & Scheule, H. (2016). Credit Risk Analytics: Measurement Techniques, Applications, and Examples in SAS. John Wiley & Sons.
- Kaushik Jais Github repository https://github.com/KaushikJais/Credit-Card-Default
- Estimation Procedures of Using Five Alternative Machine Learning Methods for Predicting Credit Card Default by Huei-Wen Teng and Michael Lee https://www.worldscientific.com/doi/abs/10.1142/S0219091519500218
- An Investigation of Credit Card Default Prediction in the Imbalanced Datasets by Talha Mahboob Alam and Kamran Shaukat https://ieeexplore.ieee.org/abstract/document/9239944/