CREDIT CARD DEFAULT PREDICTION

PROJECT 1

By:
Priyanka
Chandramohan

S.No	TOPIC	Page No.
1.	Introduction • Business Problems We are trying to solve • Main goal • About Dataset o Table1- DataSet Preview o Table 2 - Columns in Dataset	
2.	EDA & Business Implication Plot1 – Analysis on IsDefault Plot2 – Analysis on Sex Plot3 – Analysis on Education Plot4 – Analysis on Marriage Plot5 - Analysis on Age Plot6 – Heat map Observations from EDA	
3.	Date Cleaning and Preprocessing • Approach using for identifying and treating missing values and outlier treatment Code Sample 1- Null Values Code Sample 2 - Null Values % Code Sample 3 – Name Change	
4.	Model Building Model Selection and Why? o Code Sample 5- Splitting Data o Code Sample 6- List of Algo used Plot 7 – ROC CURVE Plot 8 - Feature Importance Efforts to improve model performance Hyperparameter Tuning • Code sample 4 – Best Parameter grid LR • Code sample 5 – Hyper parameter Tuning LR • Code sample 6 – Best Parameter grid DT • Code sample 7 – Hyper parameter Tuning DT • Code sample 8 – Best Parameter grid SVC • Code sample 9 – Hyper parameter Tuning SVC • Code sample 10 – Best Parameter grid XGBoost • Code sample 11 – Hyper parameter Tuning XGBoost	
5.	Final Interpretation/Recommendation Table 6 - Final Algorithm Performance Table Recommendation	

1. Introduction

Business Problems we are Trying to Solve

- This project is aimed at predicting the case of customers default payments in Taiwan. From the perspective of risk management, the result of predictive accuracy of the estimated probability of default will be more valuable than the binary result of classification - credible or not credible clients. We can use the K-S chart to evaluate which customers will default on their credit card payments

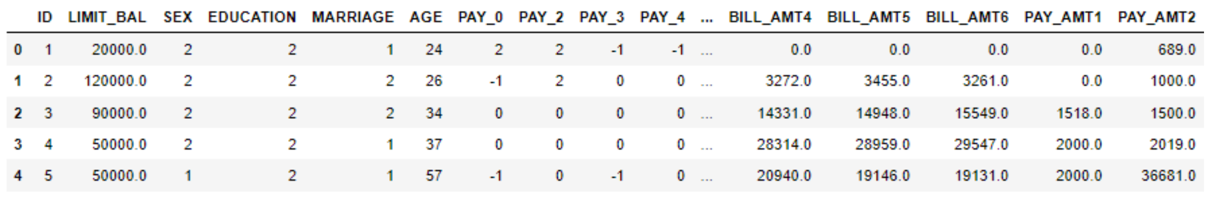
Objective:

- Predicting whether a customer will default on his/her credit card
- EDA on credit card default prediction.
- Handling class imbalance
- Fitting different models and cross validate them.

About Dataset

This dataset contains information on default payments, demographic factors, credit data, history of payment, and bill statements of credit card clients in Taiwan from April 2005 to September 2005.

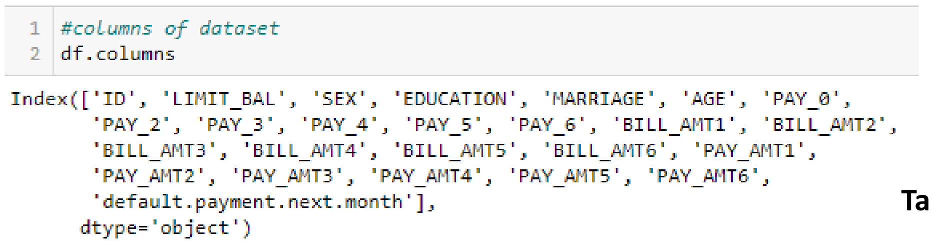
This is what the dataset looks like,



	ID	LIMIT_BAL	SEX	EDUCATION	MARRIAGE	AGE	PAY_0	PAY_2	PAY_3	PAY_4	...	BILL_AMT4	BILL_AMT5	BILL_AMT6	PAY_AMT1	PAY_AMT2
0	1	20000.0	2	2	1	24	2	2	-1	-1	...	0.0	0.0	0.0	0.0	689.0
1	2	120000.0	2	2	2	26	-1	2	0	0	...	3272.0	3455.0	3261.0	0.0	1000.0
2	3	90000.0	2	2	2	34	0	0	0	0	...	14331.0	14948.0	15549.0	1518.0	1500.0
3	4	50000.0	2	2	1	37	0	0	0	0	...	28314.0	28959.0	29547.0	2000.0	2019.0
4	5	50000.0	1	2	1	57	-1	0	-1	0	...	20940.0	19146.0	19131.0	2000.0	36681.0

Table1- Dataset

It has lots of rows and columns , which is 30000 rows spread across 25 columns. Here is the list of columns this dataset has,



```
#columns of dataset
df.columns
```

```
Index(['ID', 'LIMIT_BAL', 'SEX', 'EDUCATION', 'MARRIAGE', 'AGE', 'PAY_0',
       'PAY_2', 'PAY_3', 'PAY_4', 'PAY_5', 'PAY_6', 'BILL_AMT1', 'BILL_AMT2',
       'BILL_AMT3', 'BILL_AMT4', 'BILL_AMT5', 'BILL_AMT6', 'PAY_AMT1',
       'PAY_AMT2', 'PAY_AMT3', 'PAY_AMT4', 'PAY_AMT5', 'PAY_AMT6',
       'default.payment.next.month'],
      dtype='object')
```

Table 2 - Columns in

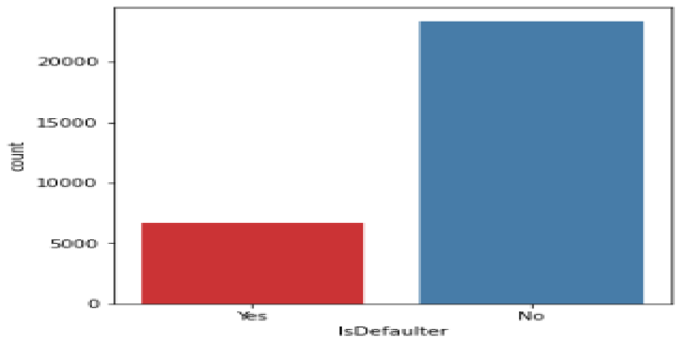
2. EDA & Business Implication

EDA stands for exploratory data analysis where we explore our data and grab insights from it. EDA helps us in getting knowledge in form of various plots and diagrams where we can easily understand the data and its features.

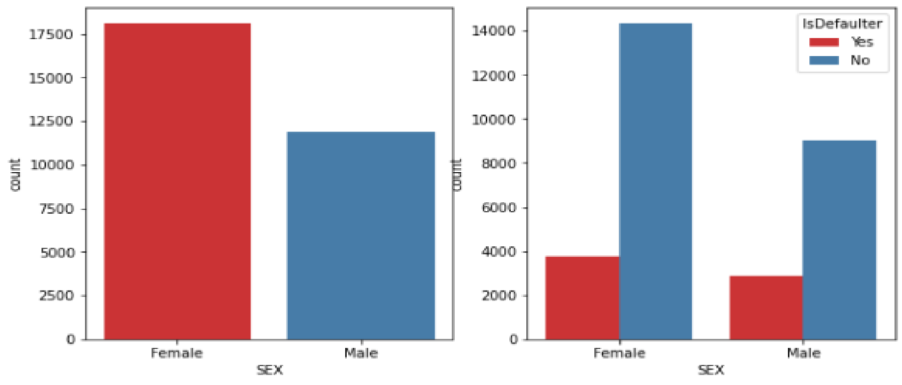
Analysis of the Data

	count	mean	std	min	25%	50%	75%	max
ID	30000.0	15000.500000	8660.398374	1.0	7500.75	15000.5	22500.25	30000.0
LIMIT_BAL	30000.0	167484.322667	129747.661567	10000.0	50000.00	140000.0	240000.00	1000000.0
SEX	30000.0	1.603733	0.489129	1.0	1.00	2.0	2.00	2.0
EDUCATION	30000.0	1.853133	0.790349	0.0	1.00	2.0	2.00	6.0
MARRIAGE	30000.0	1.551867	0.521970	0.0	1.00	2.0	2.00	3.0
AGE	30000.0	35.485500	9.217904	21.0	28.00	34.0	41.00	79.0
PAY_0	30000.0	-0.016700	1.123802	-2.0	-1.00	0.0	0.00	8.0
PAY_2	30000.0	-0.133767	1.197186	-2.0	-1.00	0.0	0.00	8.0
PAY_3	30000.0	-0.166200	1.196868	-2.0	-1.00	0.0	0.00	8.0
PAY_4	30000.0	-0.220667	1.169139	-2.0	-1.00	0.0	0.00	8.0
PAY_5	30000.0	-0.266200	1.133187	-2.0	-1.00	0.0	0.00	8.0
PAY_6	30000.0	-0.291100	1.149988	-2.0	-1.00	0.0	0.00	8.0
BILL_AMT1	30000.0	51223.330900	73635.860576	-165580.0	3558.75	22381.5	67091.00	964511.0
BILL_AMT2	30000.0	49179.075167	71173.768783	-69777.0	2984.75	21200.0	64006.25	983931.0
BILL_AMT3	30000.0	47013.154800	69349.387427	-157264.0	2666.25	20088.5	60164.75	1664089.0
BILL_AMT4	30000.0	43262.948967	64332.856134	-170000.0	2326.75	19052.0	54506.00	891586.0
BILL_AMT5	30000.0	40311.400967	60797.155770	-81334.0	1763.00	18104.5	50190.50	927171.0
BILL_AMT6	30000.0	38871.760400	59554.107537	-339603.0	1256.00	17071.0	49198.25	961664.0
PAY_AMT1	30000.0	5663.580500	16563.280354	0.0	1000.00	2100.0	5006.00	873552.0
PAY_AMT2	30000.0	5921.163500	23040.870402	0.0	833.00	2009.0	5000.00	1684259.0
PAY_AMT3	30000.0	5225.681500	17606.961470	0.0	390.00	1800.0	4505.00	896040.0
PAY_AMT4	30000.0	4826.076867	15666.159744	0.0	296.00	1500.0	4013.25	621000.0
PAY_AMT5	30000.0	4799.387633	15278.305679	0.0	252.50	1500.0	4031.50	426529.0
PAY_AMT6	30000.0	5215.502567	17777.465775	0.0	117.75	1500.0	4000.00	528666.0
default.payment.next.month	30000.0	0.221200	0.415062	0.0	0.00	0.0	0.00	1.0

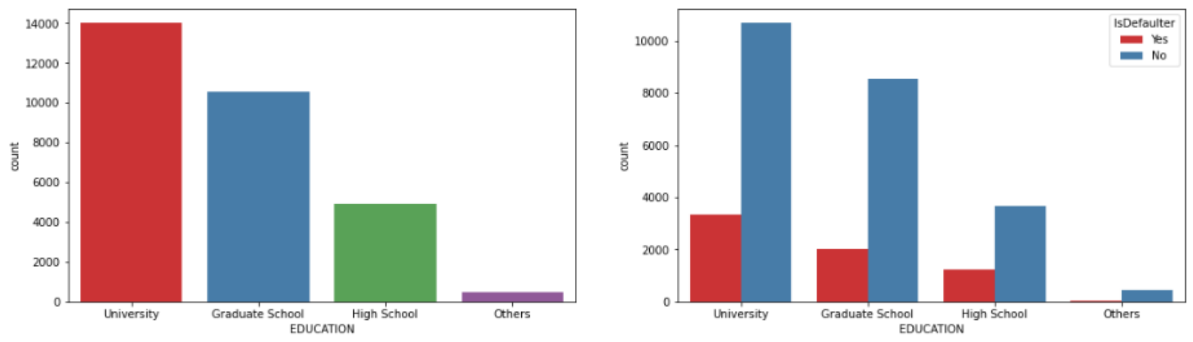
Table 3- Checking of the Skewness



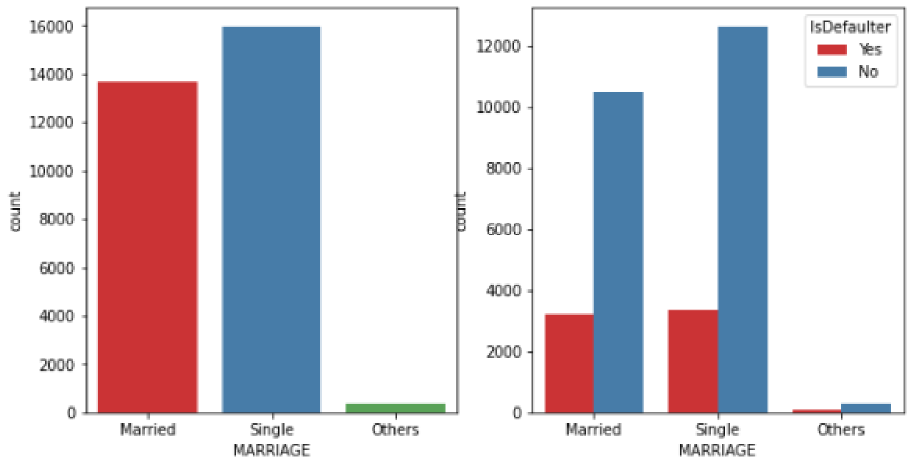
Plot 1 – Analysis on IsDefaulter



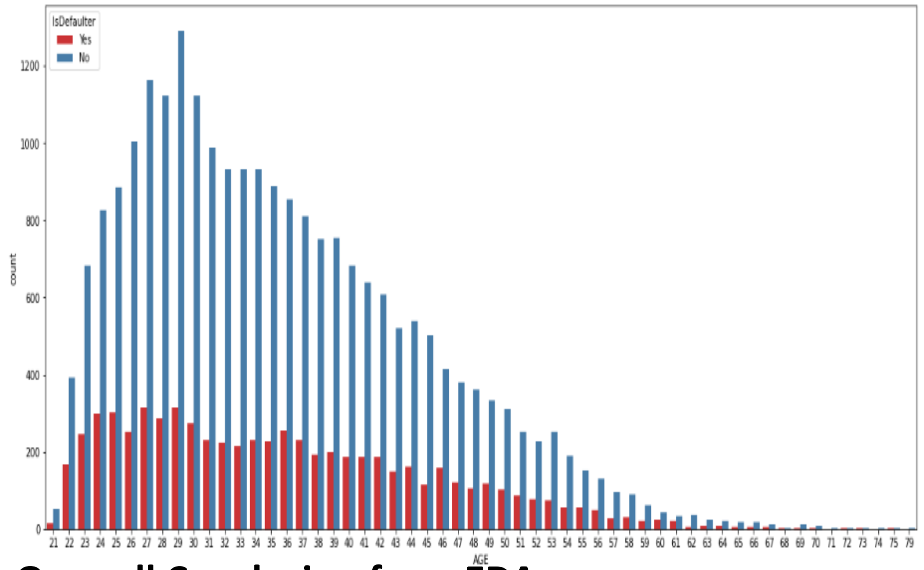
Plot 2 – Analysis on SEX



Plot 3 – Analysis on Education



Plot 4 – Analysis on Marriage

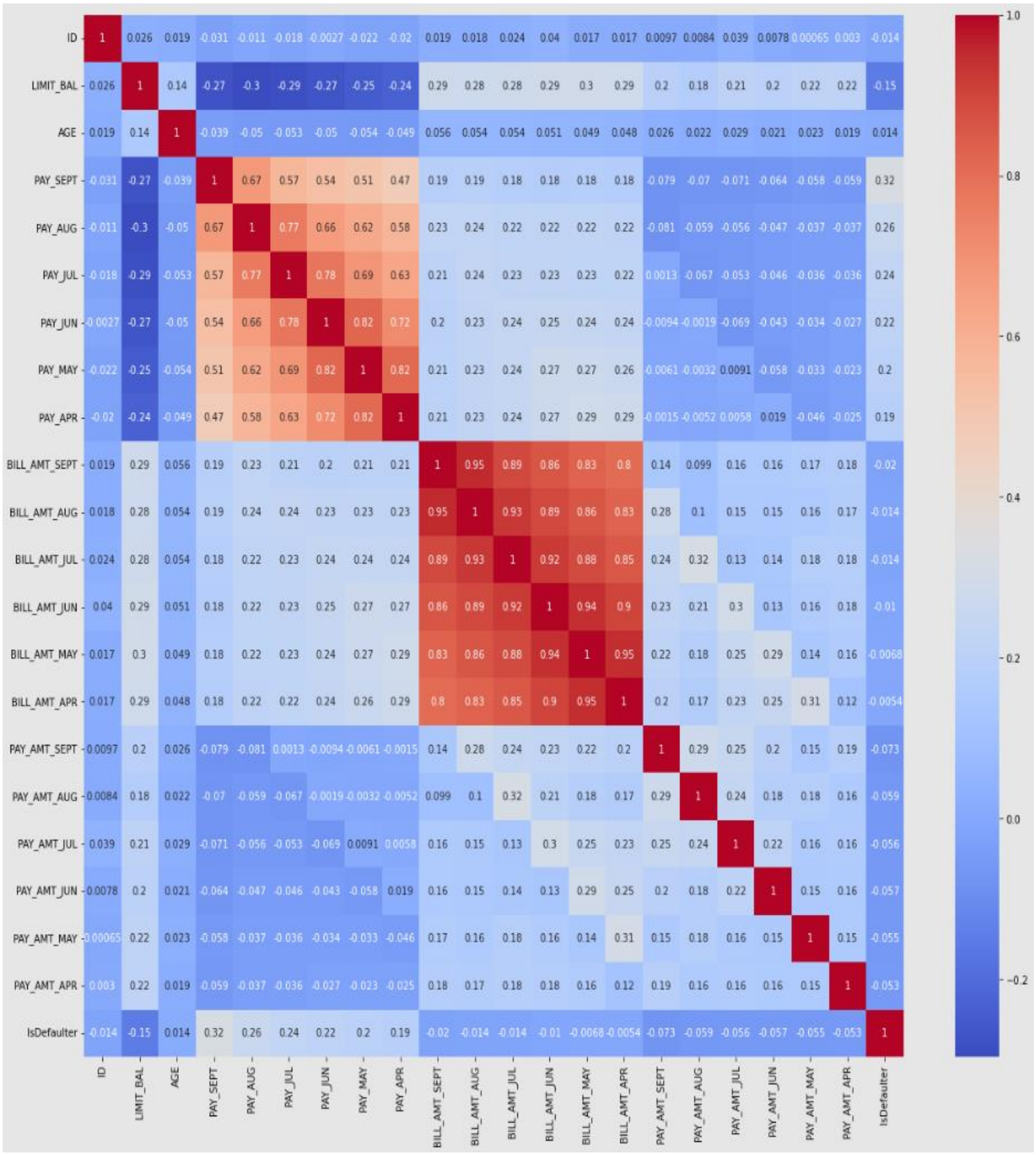


Plot 5 – Analysis on Age

Over-all Conclusion from EDA

Defaulters are less as compare to the Non-Defaulters in the given dataset. And also we can see that both classes are not in proportion that is we have an imbalanced dataset.
Female credit card holders are larger than male credit cards holders.
As the number of female credit card holder is larger than male, their credit card defaults are also higher than male.
University and graduate school has maximum credit card holder.
As the number of university and graduate school credit card holder is higher their credit card default are also higher.
Number of credit card holder is maximum in singles.
But credit card defaults are almost same in case of single and married people.

Heatmap



Plot 6 – Heatmap

3. Data Cleaning & Pre-processing

Data Cleaning is an important phase in any data science project, if our data is clean then only we can provide it to our machine learning model. Uncleaned Data can further lead our model with low accuracy. And, if data is incorrect, outcomes and algorithms are

Unreliable , even though they may look correct. There is no one absolute way to Prescribe the exact steps in the data cleaning process because the processes will vary from dataset to dataset.

The approach used for identifying and treating missing values and outlier treatment:-

Checking for null values

```
#checking for null values in dataset
df.isnull().sum()
df.isna().sum()
```

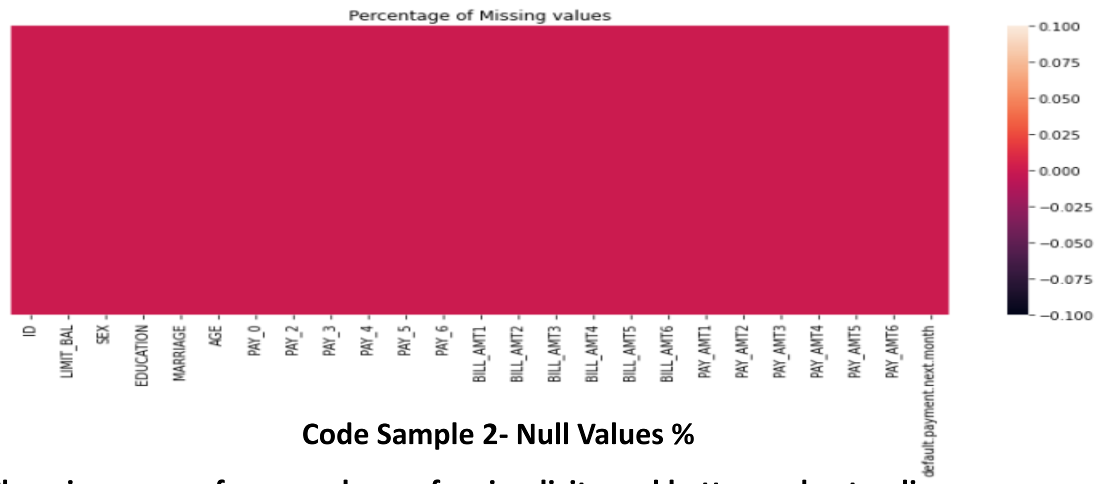
```
ID                              0
LIMIT_BAL                       0
SEX                             0
EDUCATION                       0
MARRIAGE                        0
AGE                             0
PAY_0                           0
PAY_2                           0
PAY_3                           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
```

To identify any missing values in our data set we have used Pandas pre built function isnull() to detect any missing values in our datasets. We don't have any column with null values in it. So, there is

No need for dropping or imputing the null values.

Code Sample 1- Null Values

```
#Plot to check null values
plt.figure(figsize=(14, 5))
sns.heatmap(df.isnull(), cbar=True, yticklabels=False)
plt.title("Percentage of Missing values")
plt.show()
```



Code Sample 2- Null Values %

Changing name of some columns for simplicity and better understanding

```
#renaming of columns
df.rename(columns={'default payment next month' : 'IsDefaulter'}, inplace=True)
df.rename(columns={'PAY_0':'PAY_SEPT','PAY_2':'PAY_AUG','PAY_3':'PAY_JUL','PAY_4':'PAY_JUN','PAY_5':'PAY_MAY','PAY_6':'
df.rename(columns={'BILL_AMT1':'BILL_AMT_SEPT','BILL_AMT2':'BILL_AMT_AUG','BILL_AMT3':'BILL_AMT_JUL','BILL_AMT4':'BILL_
df.rename(columns={'PAY_AMT1':'PAY_AMT_SEPT','PAY_AMT2':'PAY_AMT_AUG','PAY_AMT3':'PAY_AMT_JUL','PAY_AMT4':'PAY_AMT_JUN'
```

```
#replacing values with there labels
df.replace({'SEX': {1 : 'Male', 2 : 'Female'}}, inplace=True)
df.replace({'EDUCATION' : {1 : 'Graduate School', 2 : 'University', 3 : 'High School', 4 : 'Others'}}, inplace=True)
df.replace({'MARRIAGE' : {1 : 'Married', 2 : 'Single', 3 : 'Others'}}, inplace = True)
df.replace({'IsDefaulter' : {1 : 'Yes', 0 : 'No'}}, inplace = True)
```

Code Sample 3 – Name Change

Need for Variable Transformation:-

Variable transformation is a way to make the data work better in your model. Here specifically we have not done anything for transforming our dataset in terms of creating a new variable or anything. The need for Variable Transformation, this is because we need our model to have a good score and accuracy which will make good predictions. So to feed the data to our model we must ensure to take these steps and make our data insightful.

Variables removed or added and why?

Variables are removed in such a scenario where we have a large number of null values in any columns particularly or else in our dataset, to make our data clean and ready for

Modelling . Whereas, variables are added in such a scenario where Consider an example where we have a column as start-date and another column as end-date so we

can create a column named 'difference' where we subtract the start-date and end-date and have a number of days between them in a column named 'difference' and later drop start - date and end-date as if they are of no use.

4. Modelling Building

Model Selection and Why?

After cleaning and processing the data then comes the modeling part which includes building Machine Learning models, let's first understand in brief what Machine Learning is?

Machine Learning is a technique that analyzes past data and tries to extract meaningful insights and patterns from them which can be further used to perform predictions in future.

For example, classifying whether a tumor is benign or malignant, predicting stock prices, etc. One such application which we're using right here is predicting Credit card Default payments. Before making predictions first we need to build a model and train it using past data.

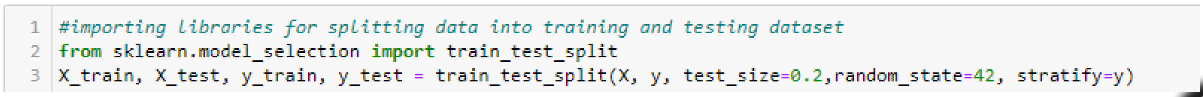
Then the data needs to be split into 3 sets

1. Training set - This will be the part of the dataset which the model will be using to train itself, the size should be at least 60-70% of the total data we've.

2. Validation set - This set is used for validating our model's performance for a different set of hyperparameters. After taking out the train set, the remaining set can be split into validation and test set.

3. Testing set - To evaluate how the model is performing on the unseen data on which the model will be doing future predictions on, test set is used. It helps to understand how much error is there between actual and predicted values.

Train Test Splitting

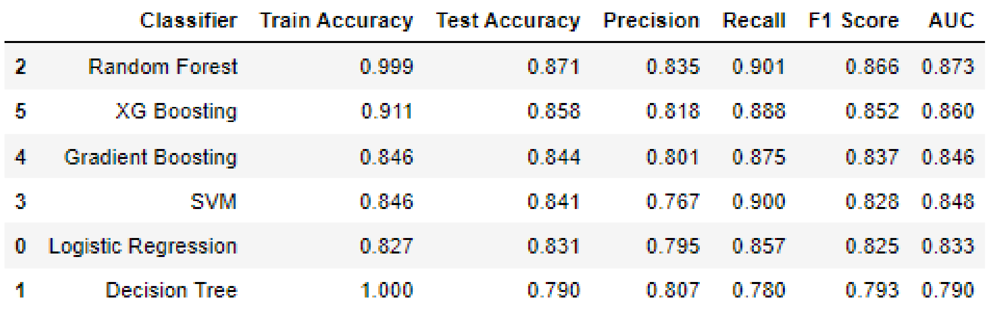


```
#importing libraries for splitting data into training and testing dataset
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,random_state=42, stratify=y)
```

We need to build different classification algorithms and using the validation set we can determine which model to keep for making final predictions.

Here is the list of all the algorithms we've to build and evaluated:

1. Logistic Regression
2. Logistic Regression with Grid Search CV
3. **SVC**
4. SVC with Random Search CV
5. Decision Tree
6. Decision Tree with Grid Search CV
7. Random Forest
8. Random Forest with Grid Search CV
9. Gradient Boosting
10. Gradient Boosting with Grid Search CV
11. XGboost
12. Xgboost with Random Search CV

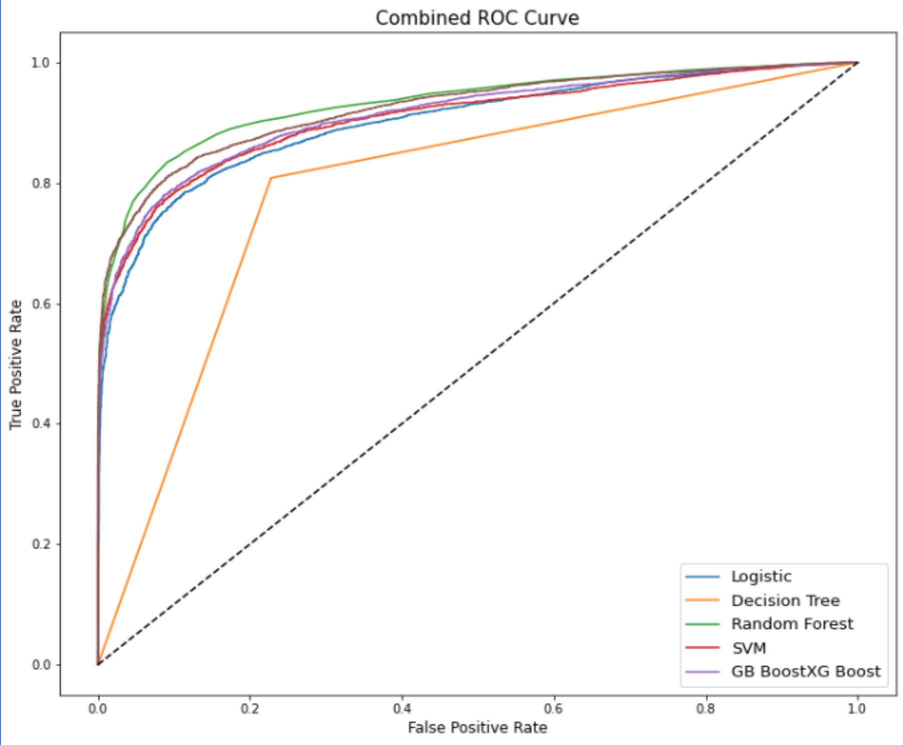


	Classifier	Train Accuracy	Test Accuracy	Precision	Recall	F1 Score	AUC
2	Random Forest	0.999	0.871	0.835	0.901	0.866	0.873
5	XG Boosting	0.911	0.858	0.818	0.888	0.852	0.860
4	Gradient Boosting	0.846	0.844	0.801	0.875	0.837	0.846
3	SVM	0.846	0.841	0.767	0.900	0.828	0.848
0	Logistic Regression	0.827	0.831	0.795	0.857	0.825	0.833
1	Decision Tree	1.000	0.790	0.807	0.780	0.793	0.790

Table 3 : Classification Result

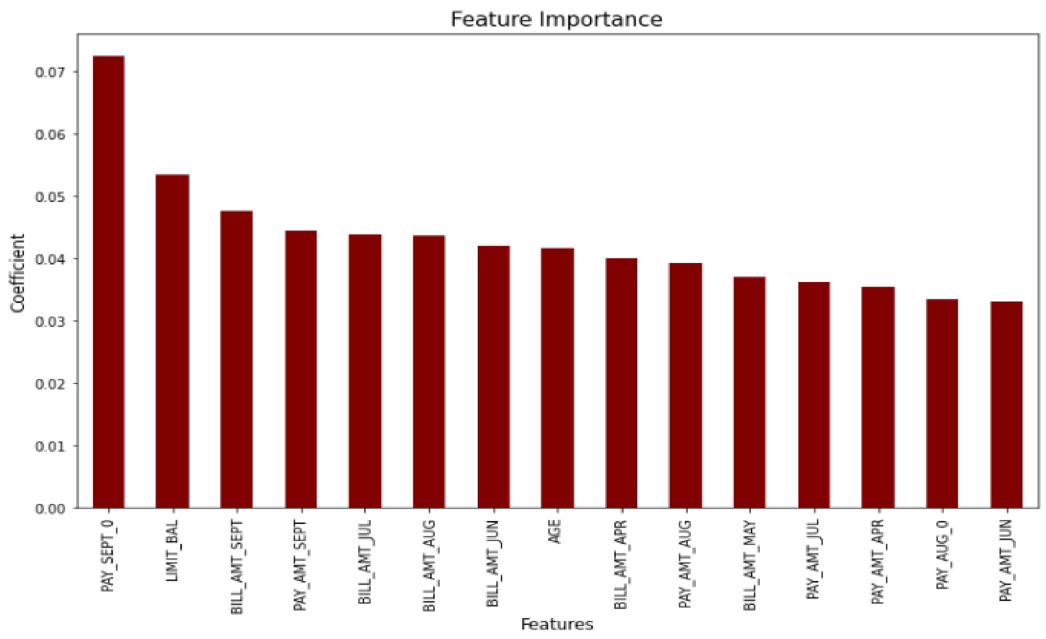
Here we can see that Random forest classifier shows highest test accuracy and F1 score.

ROC CURVE



Plot 7 - ROC AUC CURVE

Feature Importance



Above 15 features are the most important features of random forest classifier.

Plot 8 – Feature Importance

Efforts to improve model performance

Hyperparameter Tuning

Hyperparameter tuning is the process of trying out a different set of model parameters, actually, they are algorithm's parameter for example theta1 and theta2 in the hypothesis function for Linear Regression is model parameter, but lambda which is a factor that decides the amount of regularization is algorithm's parameter called as a hyperparameter.

Tuning hyperparameters helps us to find out the optimal parameter values for which the model is giving less error and a better overall score. We've performed tuning for Gradient

Boosting Regressor with both the methods RandomizedSearchCV as well as GridSearchCV. What random search does is from the given set of parameter values, it tries out n number of combinations whereas grid search builds a model with all the possible combinations from the given set of parameters and gives us the optimal values.

Grid Search CV

These are the parameters that we've given to Grid Search to find out the best one. We have performed hyper parameter tuning on the following algorithms:

1. Logistic Regression
2. SVC
3. Decision Tree
4. Random Forest
5. Gradient Boost
6. XGBoost

Logistic Regression

```python
logi_grid.best_params_
```

```
{'C': 0.1, 'max_iter': 100, 'penalty': 'l2'}
```

Code Sample 4- Best parameter grid Logistic Regression

```python
# Create an instance of the Logistic Regression
logi = LogisticRegression()

# Grid search
logi_grid = GridSearchCV(estimator=logi,
                         param_grid = param_dict,
                         cv = 5, verbose=3, n_jobs = -1, scoring='roc_auc')
# fitting model
logi_grid.fit(X_train,y_train)
```

Code Sample 5 – Hyper parameter Tuning Logistic Regression

```python
dtc_grid.best_params_
```

```
{'max_depth': 35, 'min_samples_leaf': 50, 'min_samples_split': 0.001}
```

Code Sample 6- Best parameter grid Decision Tree

```python
# Create an instance of the decision tree
dtc = DecisionTreeClassifier()

# Grid search
dtc_grid = GridSearchCV(estimator=dtc,
                        param_grid = param_dict,
                        cv = 5, verbose=3, n_jobs = -1, scoring='roc_auc')
# fitting model
dtc_grid.fit(X_train, y_train)
```

Code Sample 7 – Hyper parameter Tuning Decision Tree

```python
svm_grid.best_params_
```

```
{'kernel': 'rbf', 'C': 1}
```

Code Sample 8 - Best parameter grid SVC

```python
# Create an instance of the support vector classifier
svm=SVC(probability=True)

# Grid search
svm_grid = RandomizedSearchCV(estimator = svm, param_distributions = param_dict,
                              cv = 2, verbose=2, n_jobs = -1, scoring= 'roc_auc')
# fitting model
svm_grid.fit(X_train, y_train)
```

Code Sample 9 – Hyper parameter Tuning SVC

```python
xgb_grid.best_params_
```

```
{'n_estimators': 200,
 'min_samples_leaf': 50,
 'min_child_weight': 1,
 'max_depth': 15,
 'learning_rate': 0.05,
 'gamma': 0.2}
```

Code Sample 10 - Best parameter grid XGB

```python
# Create an instance of the RandomForestClassifier
xgb = XGBClassifier()

# Grid search
xgb_grid = RandomizedSearchCV(estimator=xgb,
                              param_distributions = param_dict,
                              n_jobs=-1, n_iter=5, cv = 3,
                              verbose=2, scoring='roc_auc')
# fitting model
xgb_grid.fit(X_train,y_train)
```

Code Sample 11 – Hyper parameter Tuning XGB

Performance Metrices

Just building is not enough, as we need to evaluate it using different metrics based on the problem we're solving. Model Validation helps us to understand how well the model is generalizing on the real-world data, the data which it has not seen during the training phase.

For classification problems the evaluation metrics we've used are:

- **Precision**
- **Recall**
- **Accuracy**
- **F-1 score**
- **AUC Score**

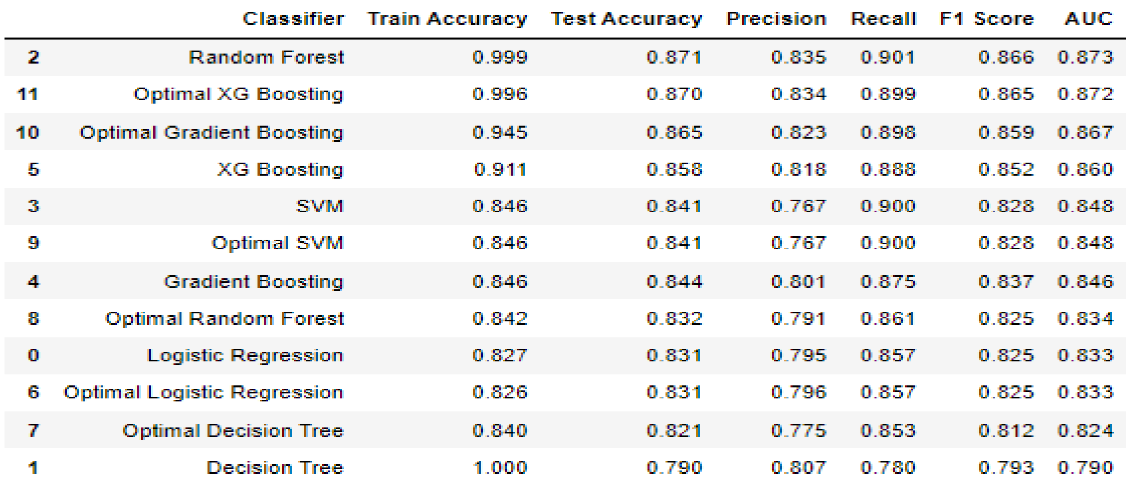
Classification accuracy, which measures the number of correct predictions made divided by the total number of predictions made, multiplied by 100 to turn it into a percentage.

$$\text{Precision} = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

$$\text{F1} = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

$$\text{Recall} = \frac{True\ Positive}{True\ Positive + False\ Negative}$$

AUC-ROC is the valued metric used for evaluating the performance in classification models. The AUC-ROC metric clearly helps determine and tell us about the capability of a model in distinguishing the classes. The judging criteria being - Higher the AUC, better the model.



	Classifier	Train Accuracy	Test Accuracy	Precision	Recall	F1 Score	AUC
2	Random Forest	0.999	0.871	0.835	0.901	0.866	0.873
11	Optimal XG Boosting	0.996	0.870	0.834	0.899	0.865	0.872
10	Optimal Gradient Boosting	0.945	0.865	0.823	0.898	0.859	0.867
5	XG Boosting	0.911	0.858	0.818	0.888	0.852	0.860
3	SVM	0.846	0.841	0.767	0.900	0.828	0.848
9	Optimal SVM	0.846	0.841	0.767	0.900	0.828	0.848
4	Gradient Boosting	0.846	0.844	0.801	0.875	0.837	0.846
8	Optimal Random Forest	0.842	0.832	0.791	0.861	0.825	0.834
0	Logistic Regression	0.827	0.831	0.795	0.857	0.825	0.833
6	Optimal Logistic Regression	0.826	0.831	0.796	0.857	0.825	0.833
7	Optimal Decision Tree	0.840	0.821	0.775	0.853	0.812	0.824
1	Decision Tree	1.000	0.790	0.807	0.780	0.793	0.790

Table – Algorithm Performance

Conclusion

- *From all baseline model, Random forest classifier shows highest test accuracy, F1 score and AUC.*
- *Baseline model of Random forest and decision tree shows huge difference in train and test accuracy which shows overfitting.*
- *After cross validation and hyperparameter tuning, XG Boost shows highest test accuracy score of 87.10% and AUC is 0.873.*
- *Cross validation and hyperparameter tuning certainly reduces chances of overfitting and also increases performance of model.*