Capstone Project-3

Credit Card Default Prediction

Aashruti A
Raneev K
Kunal M

Flow of Presentation

- Agenda
- Data Summary
- EDA
- Data Preprocessing
- ML models
- Conclusion

Agenda

- The objective of this project is t**o predict which customer might default in coming months.**
- This project aims at developing a model to predict the credit card default beforehand and to identify the potential customer base that can be offered various credit instruments so as to invite minimum default.

Data Summary

- ID: ID of each client
- credit_limit: Amount of given credit in NT dollars
- gender: Gender (1 = male, 2 = female)
- education: (1 = graduate school, 2 = university, 3 = high school, 0,4,5,6 = others)
- marital_status: Marital status (0 = others, 1 = married, 2 = single, 3 = others)*age: Age in years
- Scale for PAY_0 to PAY_6:
- (-2, -1, 0 = paid duly, 1 = payment delay for one month, 2 = payment delay for two months, ... 8 = payment delay for eight months, 9 = payment delay for nine months and above)
- payment_status_sept: Repayment status in September, 2005 (scale same as above)
- payment_status_aug: Repayment status in August, 2005 (scale same as above)
- payment_status_jul: Repayment status in July, 2005 (scale same as above)
- payment_status_jun: Repayment status in June, 2005 (scale same as above)
- payment_status_may: Repayment status in May, 2005 (scale same as above)
- payment_status_apr: Repayment status in April, 2005 (scale same as above)

Data Summary

- bill_sept: Amount of bill statement in September, 2005 (NT dollar)
- bill_aug: Amount of bill statement in August, 2005 (NT dollar)
- bill_jul: Amount of bill statement in July, 2005 (NT dollar)
- bill_jun: Amount of bill statement in June, 2005 (NT dollar)
- bill_may: Amount of bill statement in May, 2005 (NT dollar)
- bill_apr: Amount of bill statement in April, 2005 (NT dollar)
- payment_amount_sept: Amount of previous payment in September, 2005 (NT dollar)
- payment_amount_aug: Amount of previous payment in August, 2005 (NT dollar)
- payment_amount_jul: Amount of previous payment in July, 2005 (NT dollar)
- payment_amount_jun: Amount of previous payment in June, 2005 (NT dollar)
- payment_amount_may: Amount of previous payment in May, 2005 (NT dollar)
- payment_amount_apr: Amount of previous payment in April, 2005 (NT dollar)
- default_payment: Default payment, our target variable (1=yes, 0=no)
- In our dataset we got customer credit card transaction history for past 6 months, on basis of which we have to predict if customer will default or not.**

Cleaning of dataset (Null Values)

```
#printing the sum of total missing values in the dataset
print(f'Total null values are :{cred_df.isnull().sum().sum()}')

#checking missing values for every individual feature
cred_df.isnull().sum()
```

```
Total null values are :0
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
```

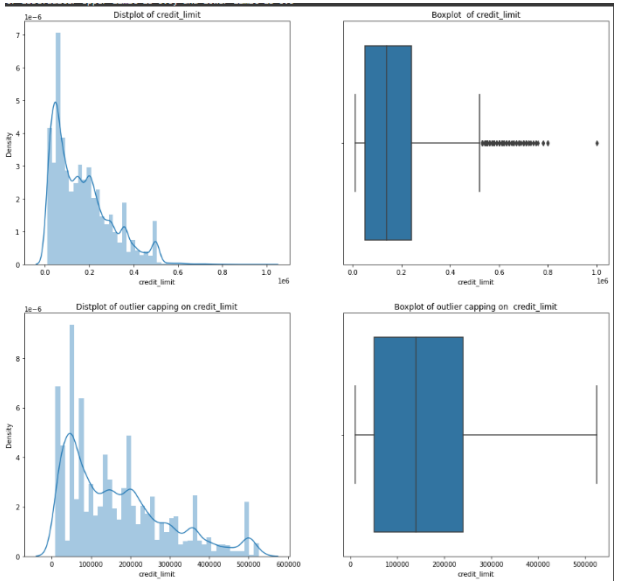
No null values

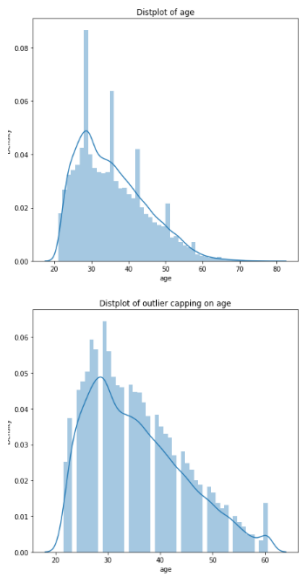
No duplicated values

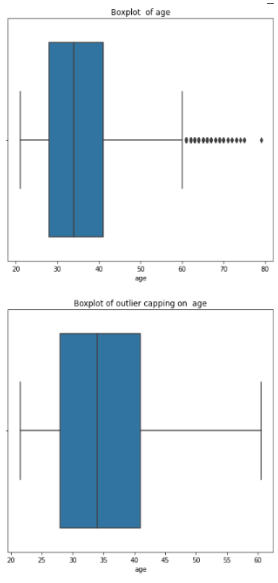
```
# Checking the duplicate values
duplicate_value = cred_df.duplicated().sum()
#value = len(cred_dfred_df.duplicated()]) #storing the total duplicated values in the dataset in a variable
print('Total duplicated values in the dataset are:', duplicate_value)
```

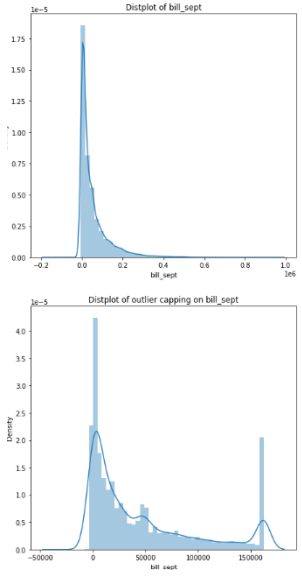
```
Total duplicated values in the dataset are: 0
```

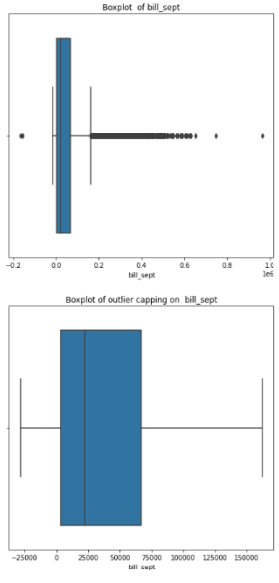
Cleaning of dataset- Outlier Handling





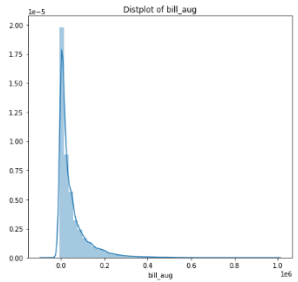


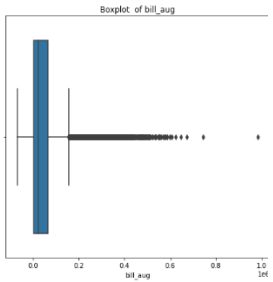


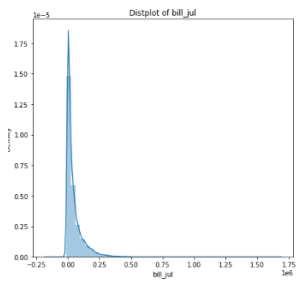


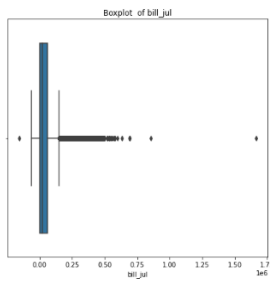
Outliers was detected using box plot and treated using IQR method

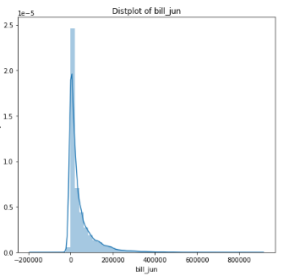
Cleaning of dataset- Outlier Handling

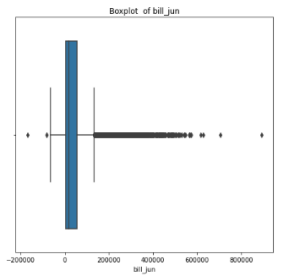


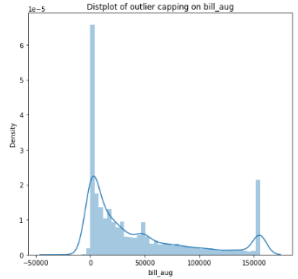


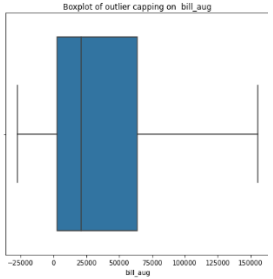


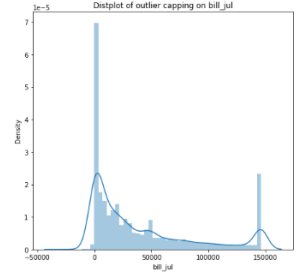


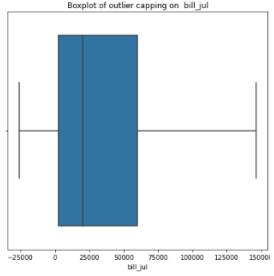


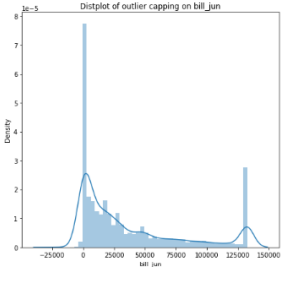


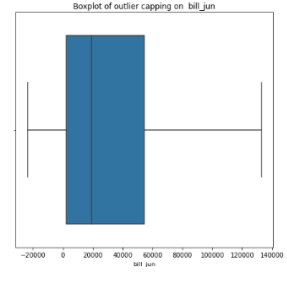




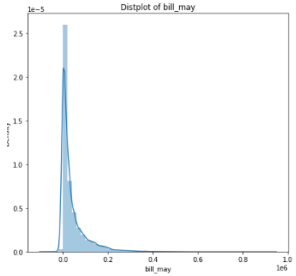


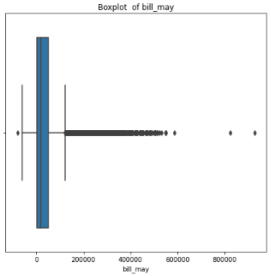


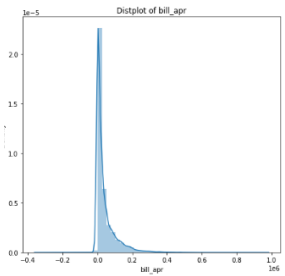


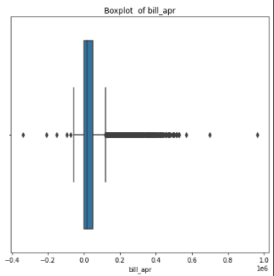


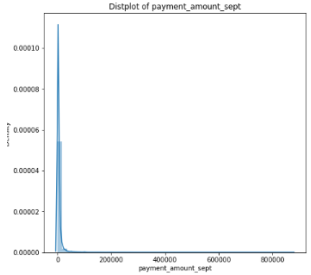
Cleaning of dataset- Outlier Handling

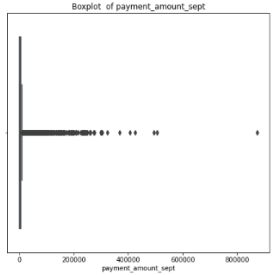


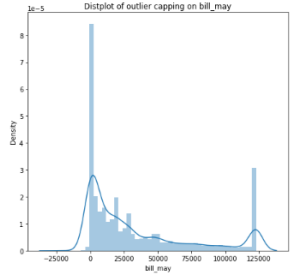


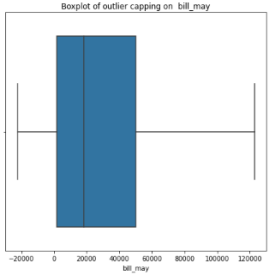


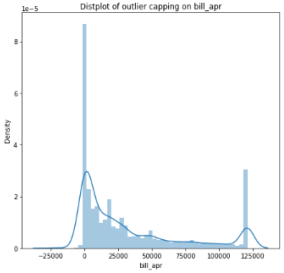


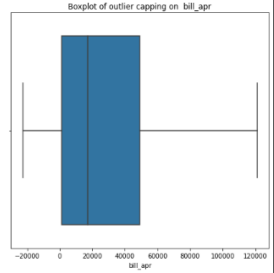


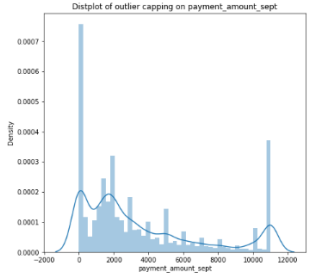


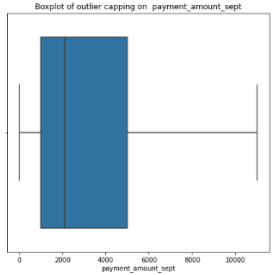




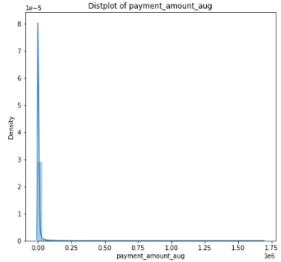


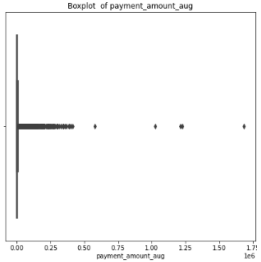


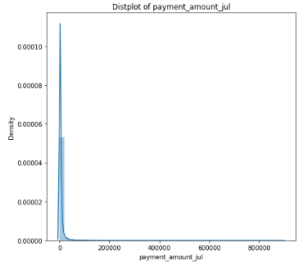


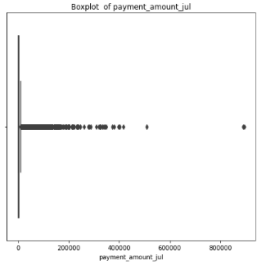


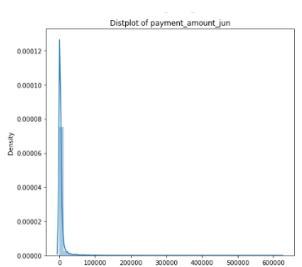
Cleaning of dataset- Outlier Handling

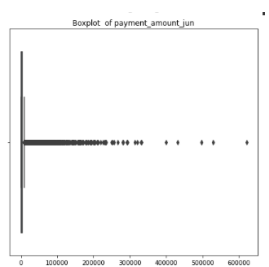


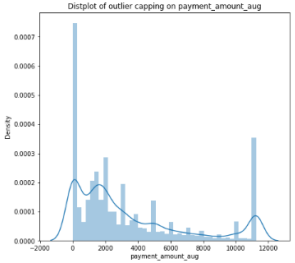


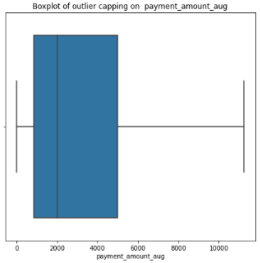


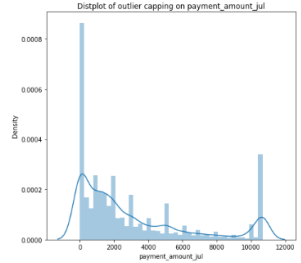


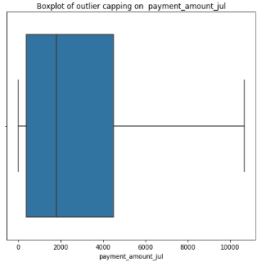


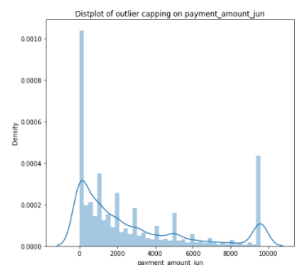


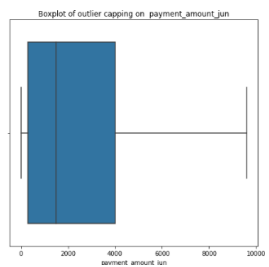




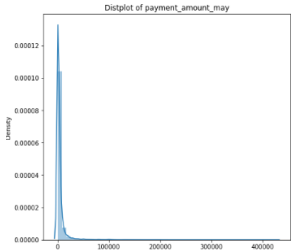


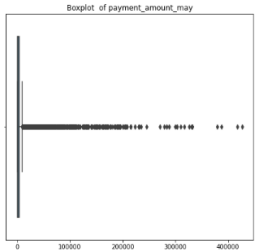


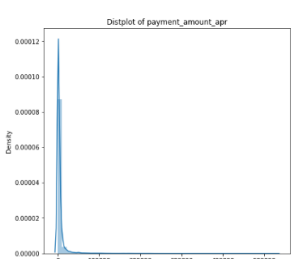


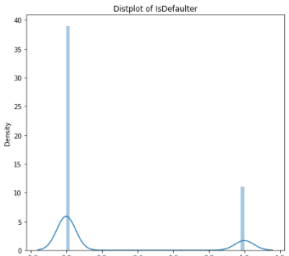


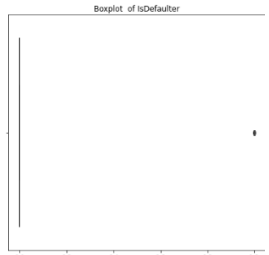
Cleaning of dataset- Outlier Handling

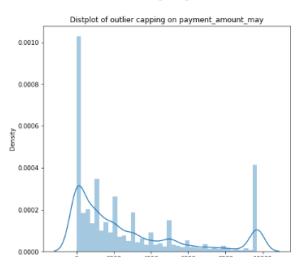


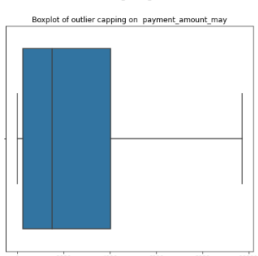


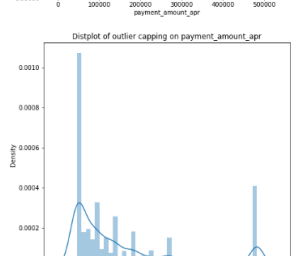


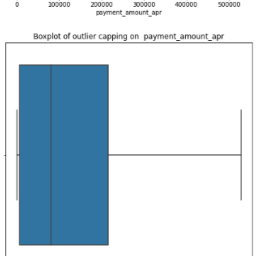


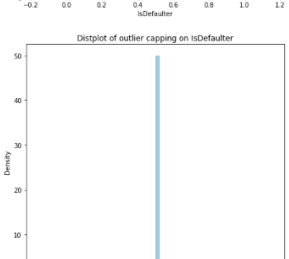


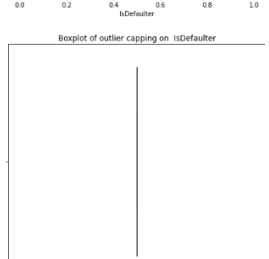




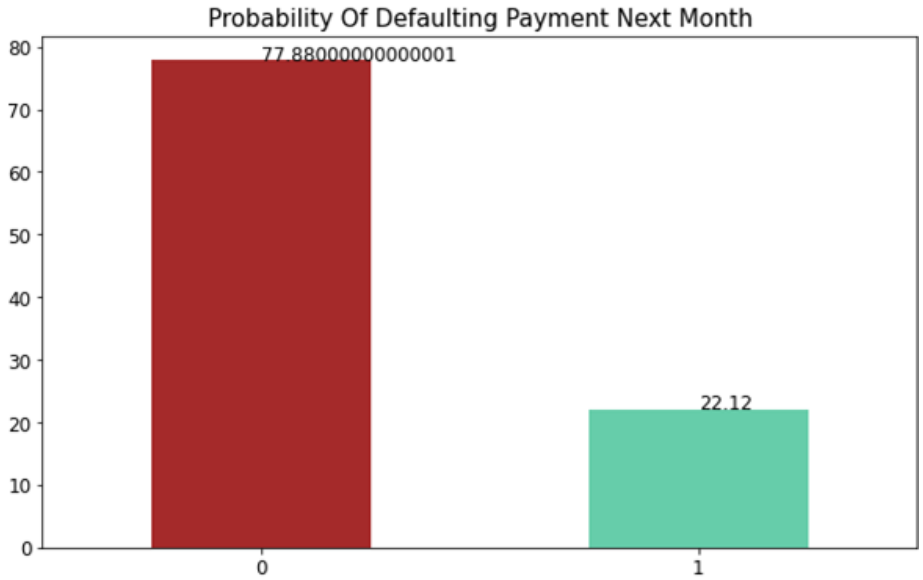






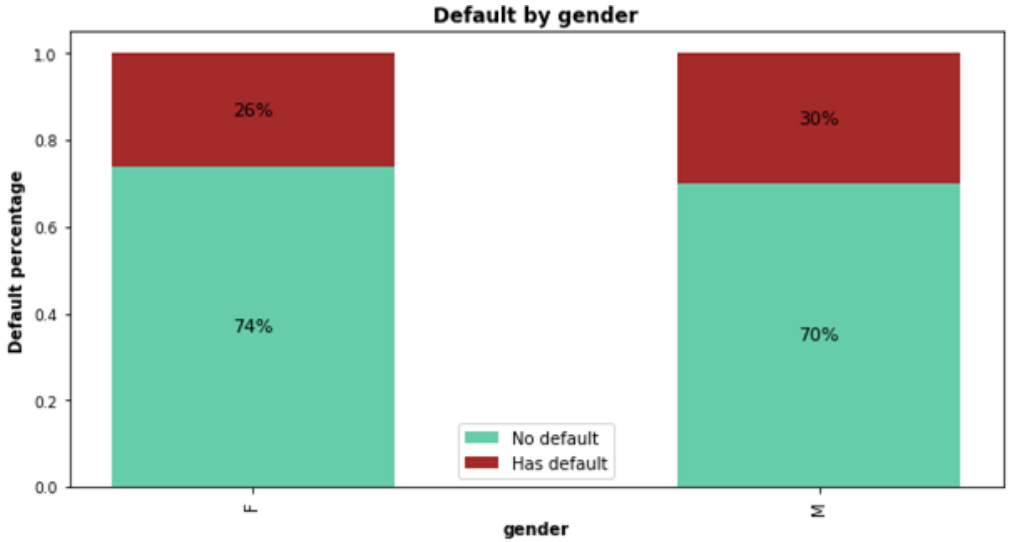


Data Visualization



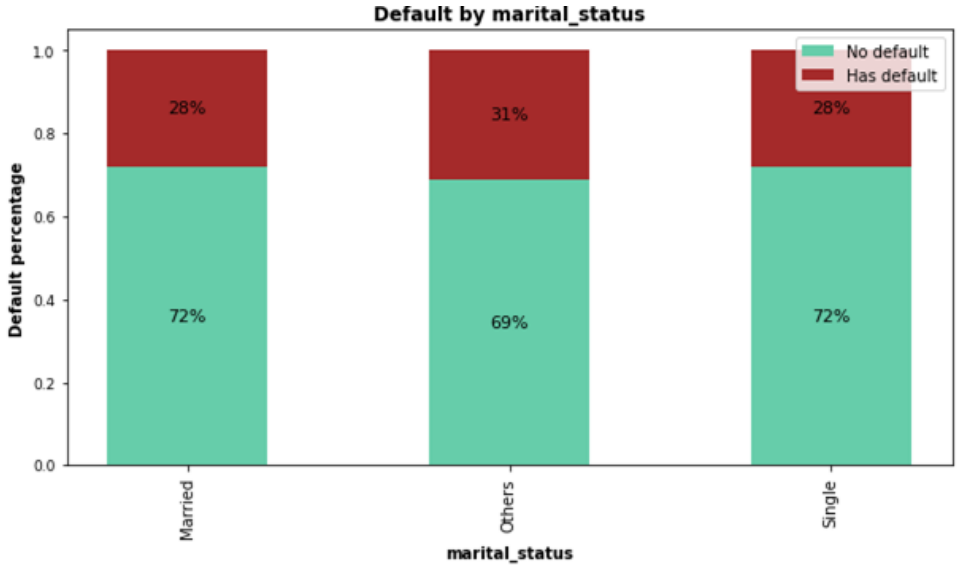
- There is a huge difference between both classes so we have to work on this **class imbalance.**
- About **22% people are expected to default next month** and 77.8% are not expected to default

Data Visualization



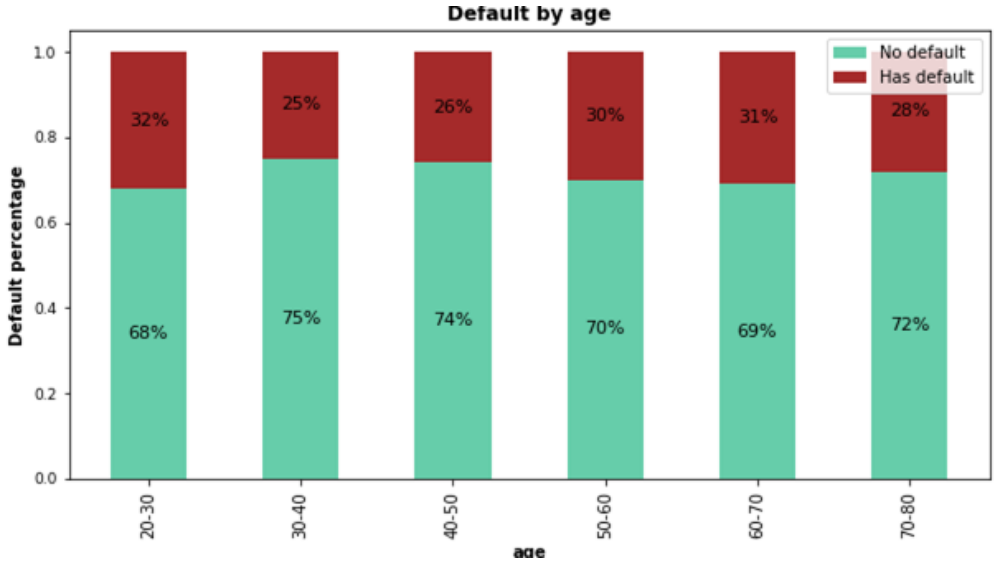
- The rate of being defaulter is **comparatively higher in males** with 30% of total defaulters than to 26% of female defaulter respectively.

Data Visualization



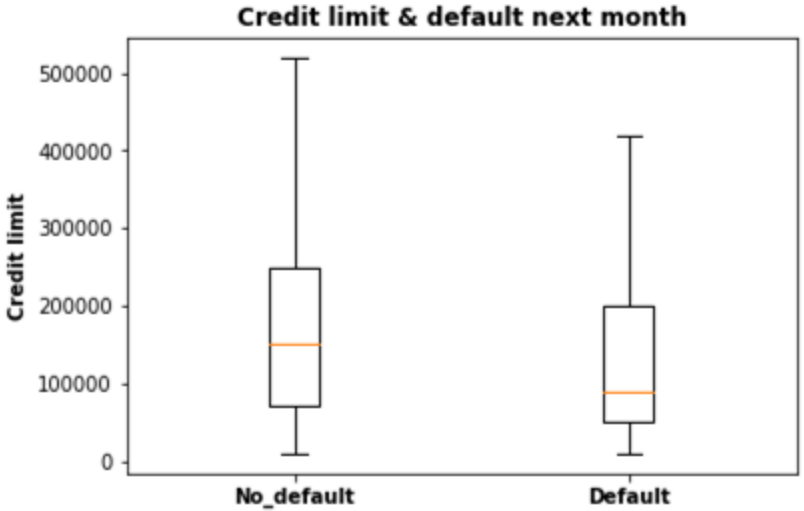
- The proportion of being defaulters is quite **consistent** across marital status categories. Marital status is not much related to the defaulter rate.

Data Visualization



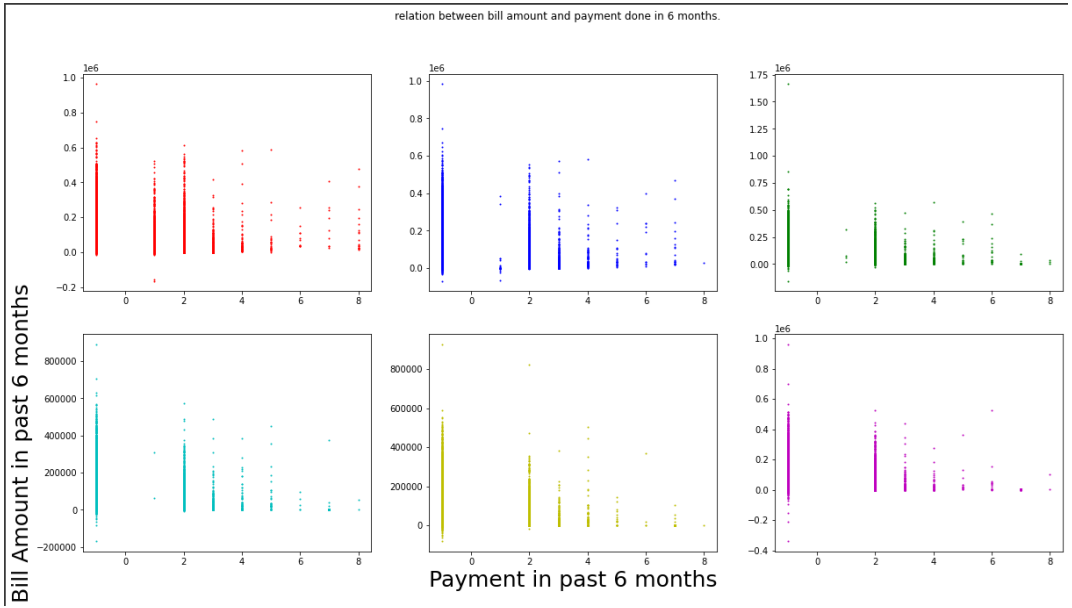
- Customers aged between **30-50 had the lowest delayed payment rate**, while younger groups (20-30) and older groups (50-70) all had higher delayed payment rates.

Data Visualization



- Customers with **high credit limits tend to pay the pay on time** and hence are not defaulters.

Data Visualization



- Above plot indicates presence of h**igher proportion of customers with higher bill amount but lower payment rate** suggesting they are likely to do the fault.
- This we can infer since maximum of data points are closely packed along the Y-axis near to 0 on X-axis.

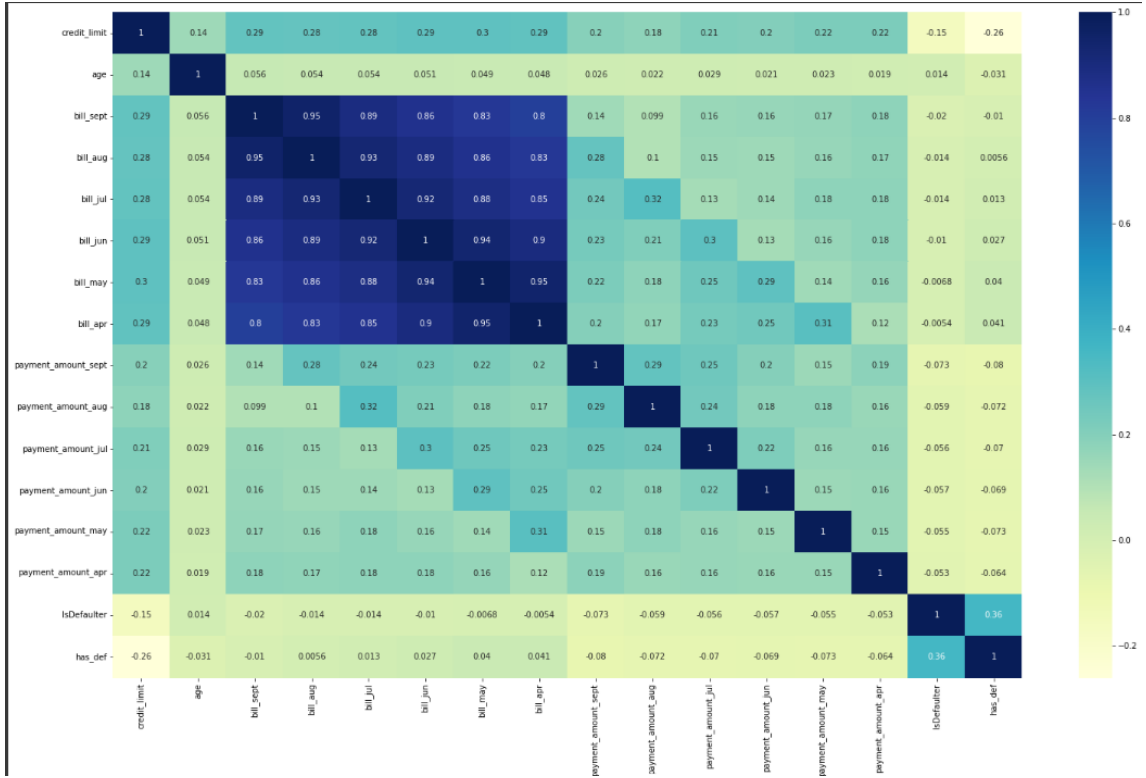
Insights- EDA

- About 22% people are expected to default next month and 77.8% are not expected to default.
- there were more female defaulter but the rate of being defaulter is comparatively higher in males with 30% of total defaulters compared to 26% of female defaulter respectively.
- there is quite similar distribution of defaulters in each category indicating marital status does not influence the defaulter customers.
- Customers with high school and university educational level had higher default percentages than customers with grad school education.
- people with higher education levels get higher credit limits.
- Maximum credit limit was 50000 NT dollars.
- Customers aged between 30-50 had the lowest delayed payment rate. They have higher credit limit and tend to ben non defaulter.

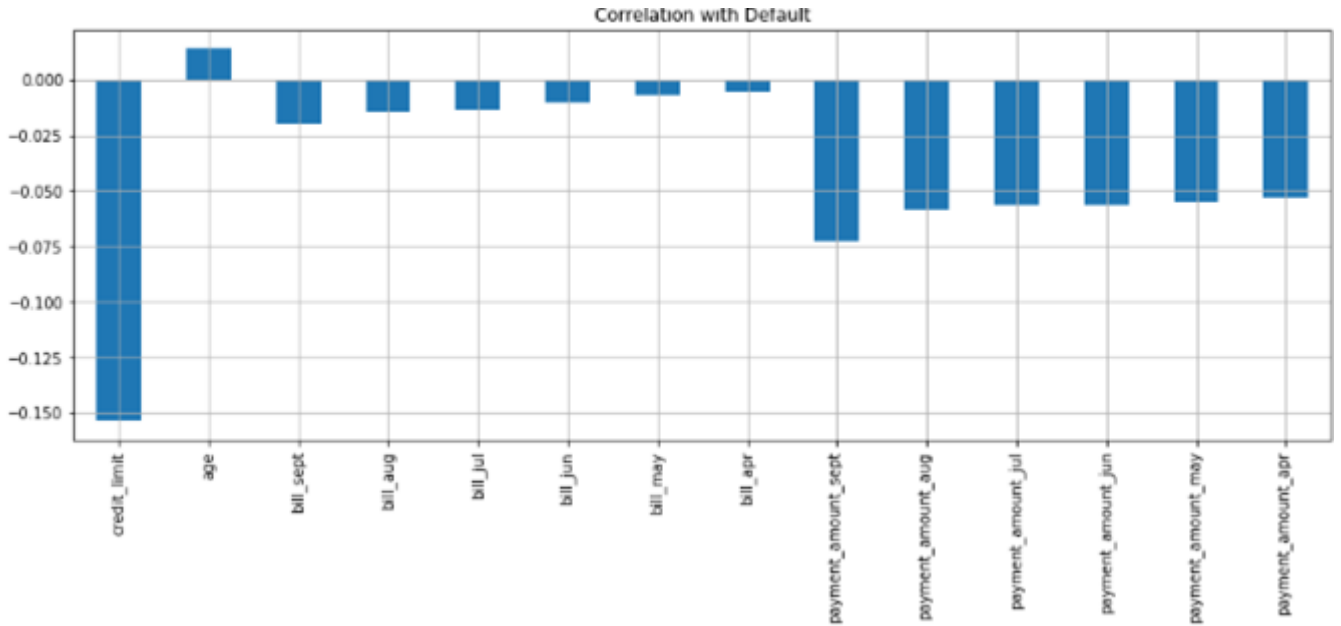
Insights- EDA

- We have negative bills in the dataset. The average negative amount is 1698 NDT which is very less when comparing with total bill amount of each month so we can guess that this would be refunds from the last billing cycles.
- the bill statement amount shouldn't exceed credit limit, however, there are 3931 customers whose bill amounts are greater than credit limit. May be the difference is due to late payment interest, assuming these customers had delayed payment.
- There are 870 customers who didn’t used their card in last 6 months.

Data Visualization- Correlation



Data Visualization



Credit_limit seems to be most negatively correlated.

Applied stats in relation with diff features to authenticate this insight.

Data Preprocessing

```
[ ]  cred_df_cpy = cred_df[['credit_limit', 'gender', 'education', 'marital_status', 'age',
           'payment_status_sept', 'payment_status_aug', 'payment_status_jul',
           'payment_status_jun', 'payment_status_may', 'payment_status_apr',
           'bill_sept', 'bill_aug', 'bill_jul', 'bill_jun', 'bill_may', 'bill_apr',
           'payment_amount_sept', 'payment_amount_aug', 'payment_amount_jul',
           'payment_amount_jun', 'payment_amount_may', 'payment_amount_apr',
           'IsDefaulter', 'bill_sum']]
```

```
[ ]  # one hot encode all the categorical features
     #Lets convert categorical fearures into object dtype first
     cred_df_cpy[['gender','marital_status','education','payment_status_sept', 'payment_status_aug', 'payment_status_jul',
           'payment_status_jun', 'payment_status_may', 'payment_status_apr']] = cred_df_cpy[['gender','marital_status','education','payment_status_sept', 'payment_status_aug', 'payment_status_jul',
           'payment_status_jun', 'payment_status_may', 'payment_status_apr']].astype('object')

     #One Hot encoding
     cred_df_cpy = pd.get_dummies(cred_df_cpy)
     cred_df_cpy.head()
```

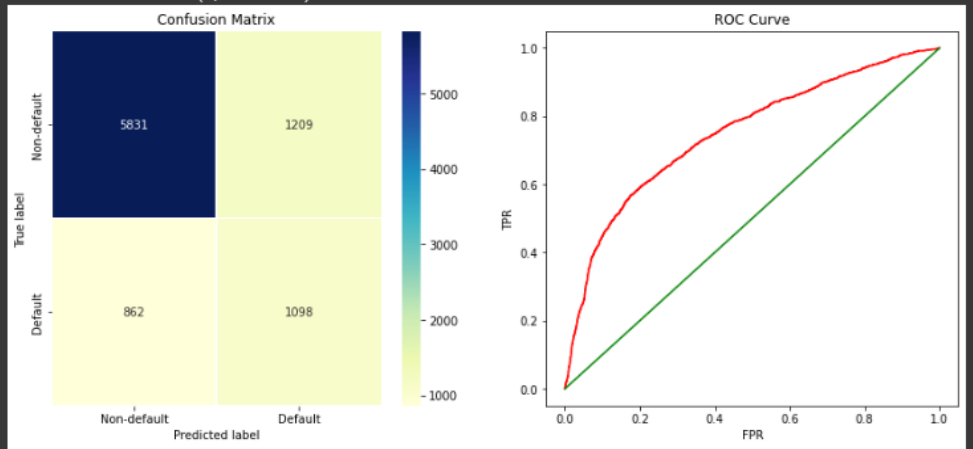
- Creating dummy variables for categorical variables using one hot encoding.
- A one hot encoding allows the representation of categorical data to be more expressive.
- Many machine learning algorithms cannot work with categorical data directly.
- The categories must be converted into numbers.
- This is required for both input and output variables that are categorical

Models used

- Logistic Regression
- Decision Tree
- Random Forest
- K Nearest Neighbor

Evaluation Metrics - Logistic Regression Model

```
Data is  SMOTE  And with hyper parameter  {'C': 0.0007196856730011522}
Overall Train Accuracy 0.7798946336682185
Train AUC Score 0.8325019344550721
Overall Train recall 0.7223719676549866
Overall Test Accuracy 0.7698888888888888
Test AUC Score 0.7495129507769016
Classification Report of Test
              precision    recall  f1-score   support

           0       0.87      0.83      0.85      7040
           1       0.48      0.56      0.51      1960

    accuracy                           0.77      9000
   macro avg       0.67      0.69      0.68      9000
weighted avg       0.79      0.77      0.78      9000

K-Fold scores: 0.779 (+/- 0.00002)
```

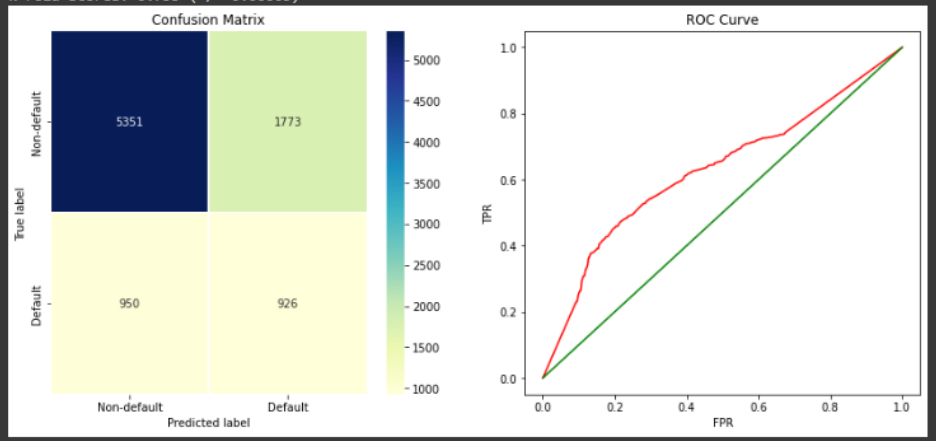


HYPERPARAMETER

```
'C': 0.0007196856730011
```

Evaluation Metrics - Decision Trees

```
Data is  SMOTE  And with hyper parameter  {'criterion': 'gini', 'max_depth': 17, 'min_samples_leaf': 1, 'min_samples_split': 8}
Overall Train Accuracy 0.8769704656640696
Train AUC Score 0.9572651938138529
Overall Train recall 0.8582472670169716
Overall Test Accuracy 0.6974444444444444
Test AUC Score 0.6291329632618171
Classification Report of Test
              precision    recall  f1-score   support

           0       0.85      0.75      0.80      7124
           1       0.34      0.49      0.40      1876

    accuracy                           0.70      9000
   macro avg       0.60      0.62      0.60      9000
weighted avg       0.74      0.70      0.72      9000

K-Fold scores: 0.733 (+/- 0.00003)
```

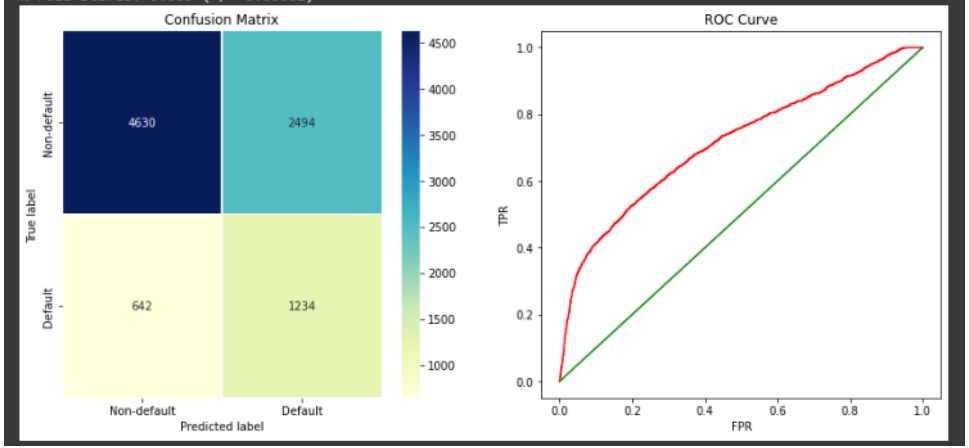


HYPERPARAMETER

```
'criterion': 'gini',
'max_depth': 17,
'min_samples_leaf': 1,
'min_samples_split': 8
```

Evaluation Metrics - Random Forest

```
Data is  SMOTE  And with hyper parameter  {'criterion': 'gini', 'max_depth': 4, 'max_features': 1, 'min_sampl
Overall Train Accuracy 0.6885305308932778
Train AUC Score 0.7621637200100451
Overall Train recall 0.7238630186628012
Overall Test Accuracy 0.6515555555555556
Test AUC Score 0.7156332643552112
Classification Report of Test
              precision    recall  f1-score   support

           0       0.88      0.65      0.75      7124
           1       0.33      0.66      0.44      1876

    accuracy                           0.65      9000
   macro avg       0.60      0.65      0.59      9000
weighted avg       0.76      0.65      0.68      9000

K-Fold scores: 0.683 (+/- 0.00002)
```

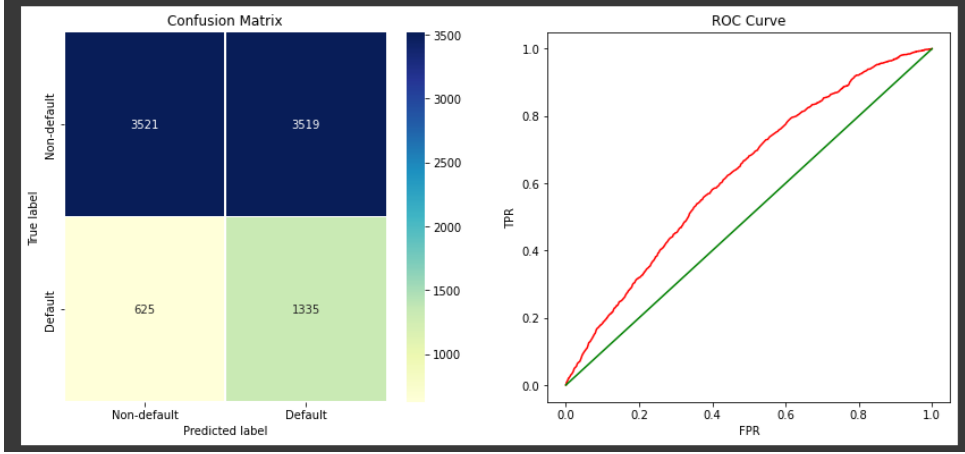


HYPERPARAMETER

```
'criterion': 'gini'
'max_depth': 4,
'max_features': 1,
'min_samples_leaf': 5,
'min_samples_split': 11,
'n_estimators': 90
```

Evaluation Metrics - KNN Algorithm

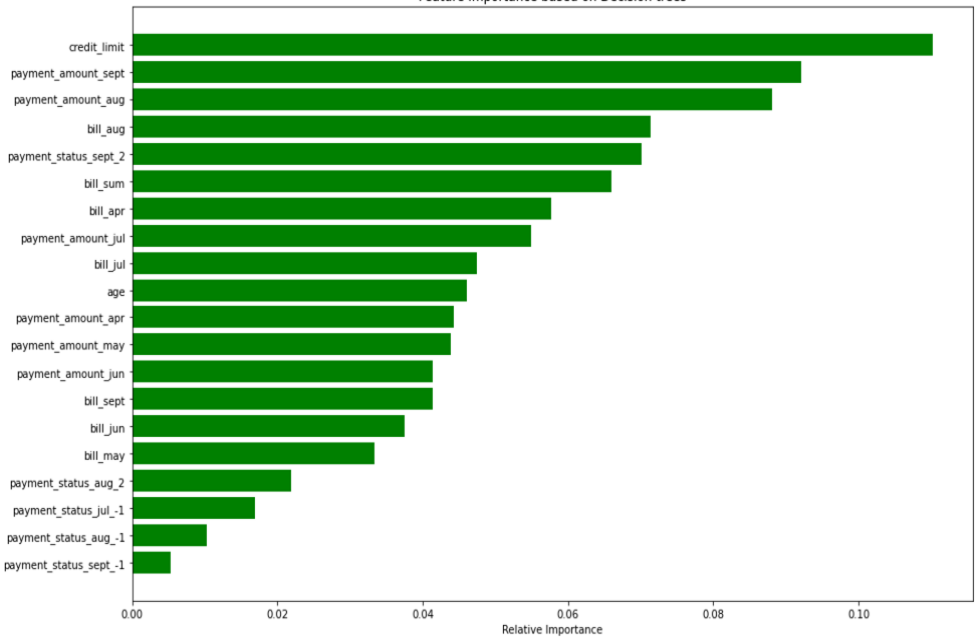
```
Data is  SMOTE sampling  And with hyper parameter  {'n_neighbors': 2000, 'leaf_size': 20}
Overall Train Accuracy 0.5983214898309238
Train AUC Score 0.6430440342288217
Overall Train recall 0.7009311443273707
Overall Test Accuracy
 0.5395555555555556
Test AUC Score
 0.6248486781076068
Classification Report of Test
              precision    recall  f1-score   support

           0       0.85      0.50      0.63      7040
           1       0.28      0.68      0.39      1960

    accuracy                           0.54      9000
   macro avg       0.56      0.59      0.51      9000
weighted avg       0.72      0.54      0.58      9000
```

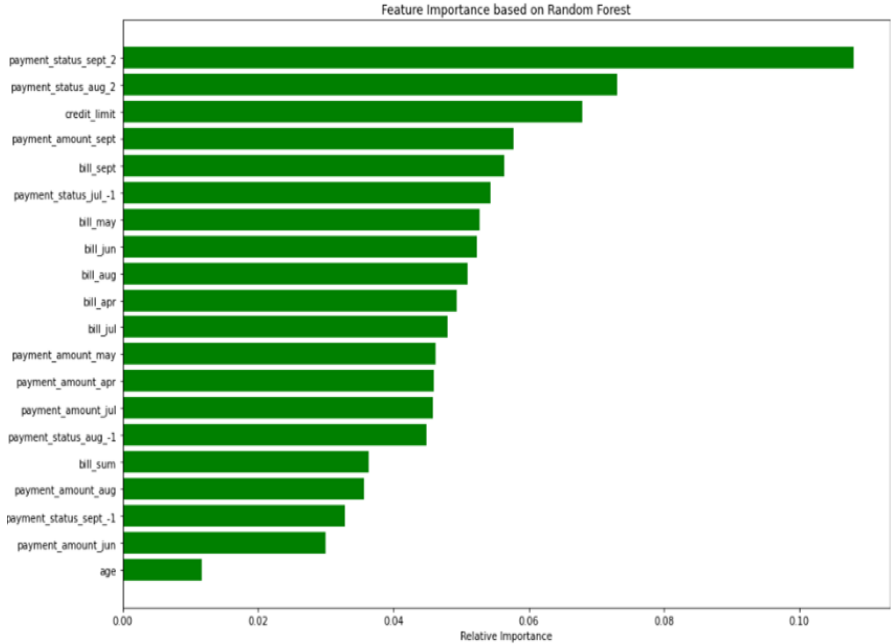


HYPERPARAMETER

```
'n_neighbour' :'2000'
'leaf_size': '20'
```

Feature Importance – Tree based model





Credit limit has given highest importance comparatively

Summary

Classifier	Train Accuracy	Test Accuracy	Precision Score	Recall Score	F1 Score
Logistic Regression	0.7798	0.7698	0.87(class 0) 0.48(class 1)	**0.83(class 0)** **0.56(class 1)**	0.85(class 0) 0.51(class 1)
Decision Tree	0.8769	0.6974	0.85(class 0) 0.34(class 1)	0.75(class 0) 0.49(class 1)	0.80(class 0) 0.40(class 1)
Random Forest	0.6885	0.6515	0.88(class 0) 0.33(class 1)	0.65(class 0) 0.66(class 1)	0.75(class 0) 0.44(class 1)
KNN Classifier	0.5983	0.5395	0.85(class 0) 0.28(class 1)	0.50(class 0) 0.68(class 1)	0.63(class 0) 0.39(class 1)

Conclusion

- There was an **imbalance** in the target variable which was balanced using **SMOTE(**Synthetic Minority Oversampling Technique).
- Logistic Regression, Decision Trees, Random Forest algorithms were implemented. The important metric to compare all the algorithms in this case is **'Recall'**.
- Logistic Regression had an imbalance in the recall score of about 83% for class 0 and 56% for class 1. Decision Trees and Random Forest have best recall scores of 75%(class 0) , 49%(class 1) and 65%(class 0), 66%(class 1), KNN classifier have recall score of 50% , 68% respectively for class 0 and class 1.
- The features like credit limit , payment amount and bill amount are important features as per Random Forest and Decision tree algorithm. Logistic Regression was performed best out of all implemented model.

THANK YOU