JULY 2022

TEXAS
The University of Texas at Austin

CREDIT SCORE CLASSIFICATION

Meet the Group -Group 8

Soumya
Agrawal

Aashi
Aashi

Kyle
Tobia

Saurabh
Arora

Ankit
Muthiyan

AGENDA FOR THE DAY

TEXAS
The University of Texas at Austin

Topics to be discussed

- Problem Statement
- Dataset Summary and Description
- Exploratory Data Analysis
- Data / Outliers Treatment
- Exploratory Data Analysis Post Data Treatment
- Modeling
- Model Comparison

PROBLEM STATEMENT

TEXAS
The University of Texas at Austin

PROBLEM STATEMENT | Given a person's credit-related information, build a machine learning model that can classify the credit score.

A global finance company has collected basic bank details and gathered a lot of credit-related information. The objective is to build an intelligent system to segregate the people into credit score brackets to reduce the manual efforts.

Data Source:

Kaggle - https://www.kaggle.com/datasets/parisrohan/credit-score-classification

DATASET SUMMARY & DESCRIPTION

DATA SUMMARY | There are 100k records with 27 fields in the dataset comprising of financial details of the individual across every month.

- ~ 100k records in the Kaggle data set
- Comprises of categorical and numerical variables
- Level of Data: Customer_ID, Month
- Highly Skewed Numerical Data; mostly right skewed
- Missing Values across multiple columns in the dataset
- Target variable has 3 class:
    - **Poor**
    - **Standard**
    - **Good**

DATA DESCRIPTION | Description of 27 fields present in the Credit Score Data gathered from Kaggle

ID- Represents a unique identification of an entry

CUSTOMER_ID – Represents a unique id of a person

MONTH - Represents the month of the year

NAME - Represents the name of a person

AGE - Represents the age of the person

SSN - Represents the social security number of a person

Occupation - Represents the occupation of the person

ANNUL_INCOME - Represents the annual income of a person

NUMBER_BANK_ACCOUNTS - Represents the number of bank accounts a person holds

Num_Credit_Card - Represents the number of other credit cards held by a person

Interest_Rate – Represents the interest rate on credit card

Num_of_Loan - the number of loans taken from the bank

Type_of_Loan - Represents the types of loan taken by a person

Delay_from_due_datesort - Represents the average number of days delayed from the payment date

Num_of_Delayed_Payment - Represents the average number of payments delayed by a person

Changed_Credit_Limit - Represents the percentage change in credit card limit

Num_Credit_Inquiries - Represents the number of credit card inquiries

Credit_Mix - Represents the classification of the mix of credits

Outstanding_Debt - Represents the remaining debt to be paid (in USD)

Credit_Utilization_Ratio - Represents the utilization ratio of credit card

Credit_History_Age - Represents the age of credit history of the person

Payment_of_Min_Amount - Represents whether only the minimum amount was paid by the person

Total_EMI_per_month - Represents the monthly EMI payments (in USD)

Amount_invested_monthly - Represents the monthly amount invested by the customer (in USD)

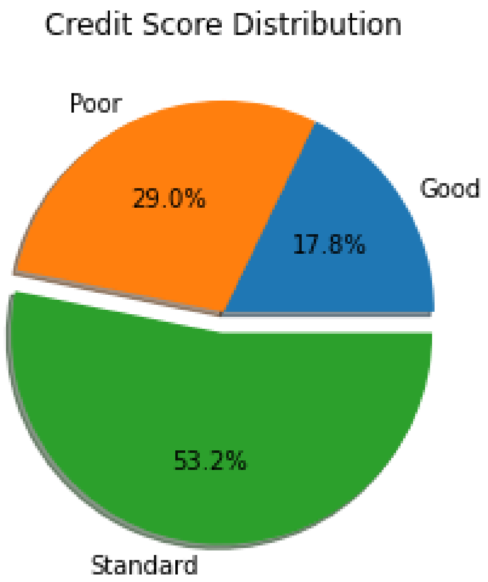
Payment_Behaviour - Represents the payment behavior of the customer (in USD)

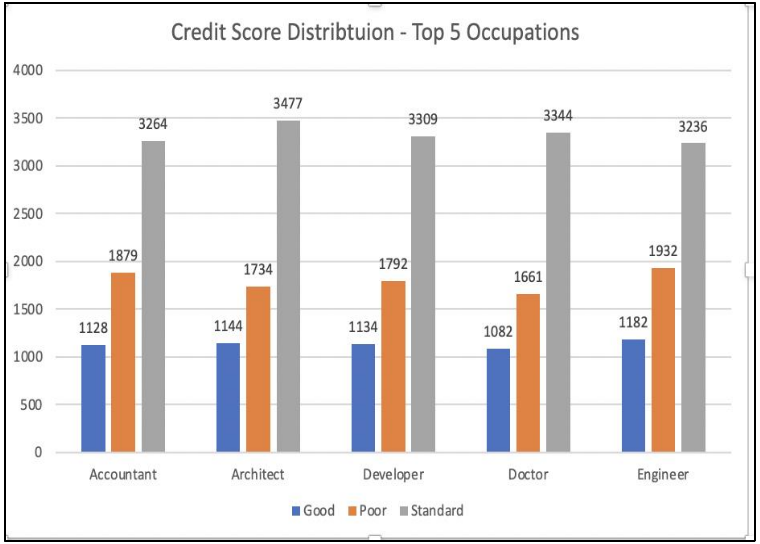
Monthly_Balance - Represents the monthly balance amount of customer (in USD)

TEXAS
The University of Texas at Austin

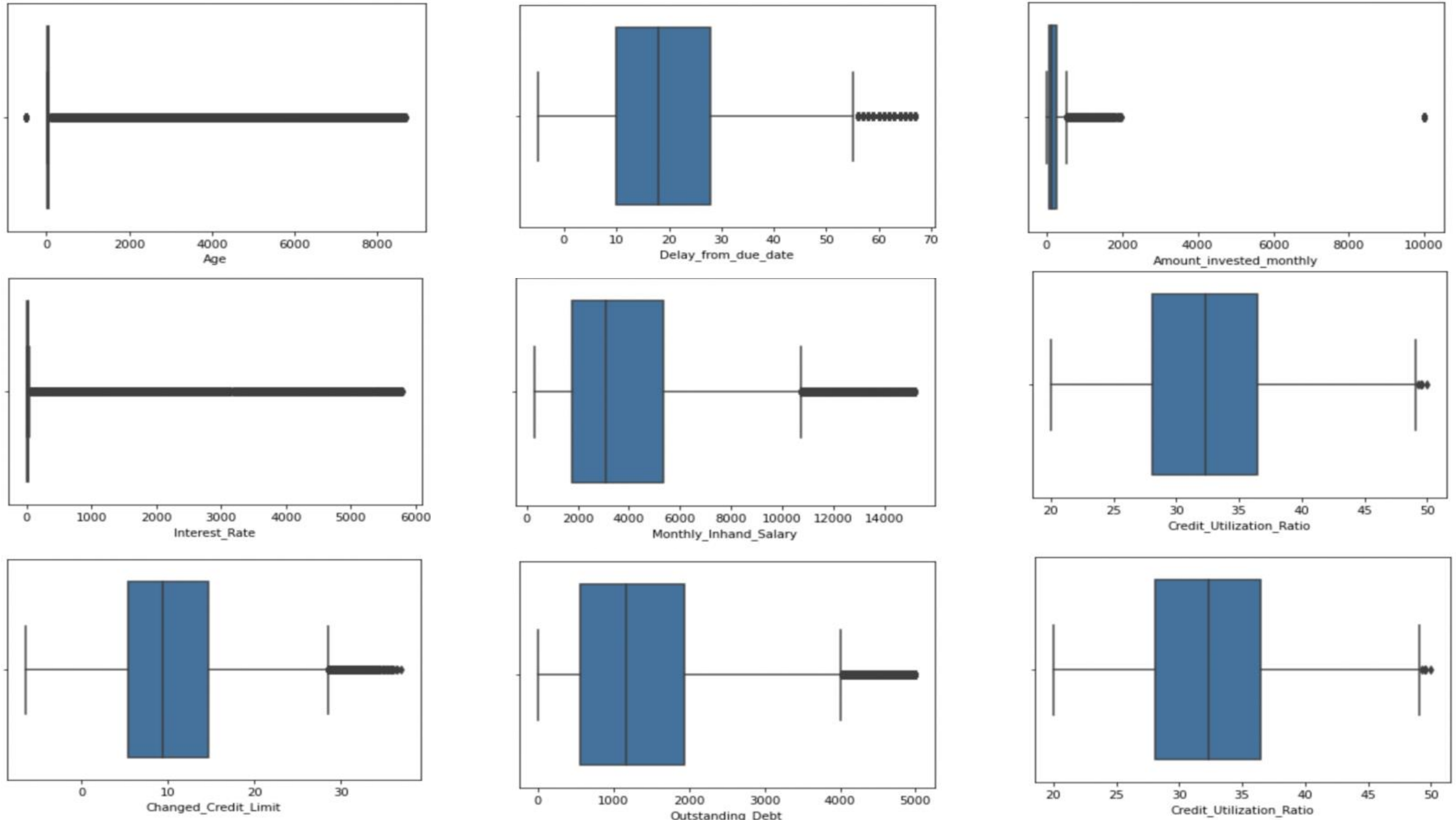
EXPLORATORY DATA ANALYSIS

EDA | Distribution of Credit Score over the dataset ; Distribution of Credit Score across Top 5 Occupation Sector

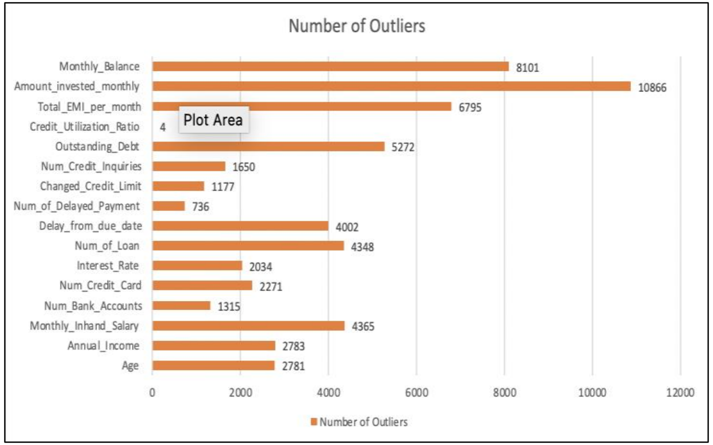


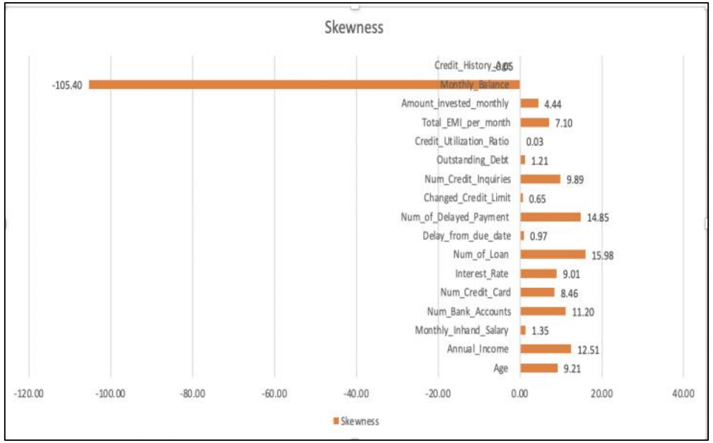


EDA | Uni-Variate Analysis | Boxplots for Numerical Variables to see the distribution along with to detect outliers present in data



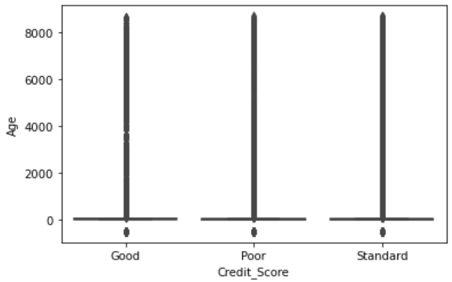
EDA | Uni-Variate Analysis | Checking the Number of Outliers in the dataset along the Skewness of the Numerical columns

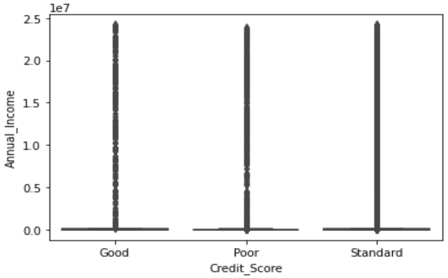


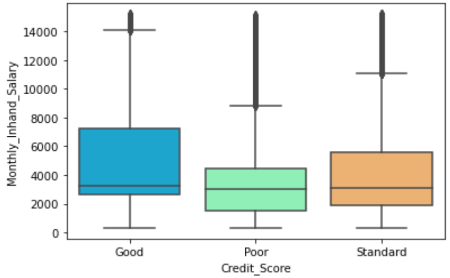


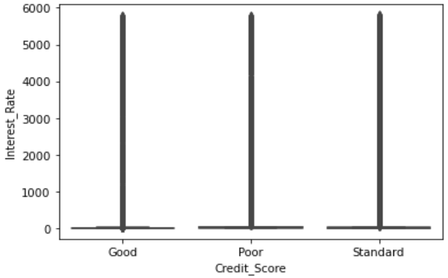
As observed, data is highly skewed, especially towards the right side; Also, the number of Outliers for each of the measure column is high, this might cause weird behavior in the Classification Model

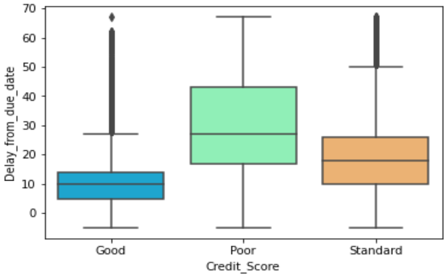
EDA | Bi-Variate Analysis | Box-Plots for Numerical Variables with respect to different Credit Score Categories Before Outlier Treatment

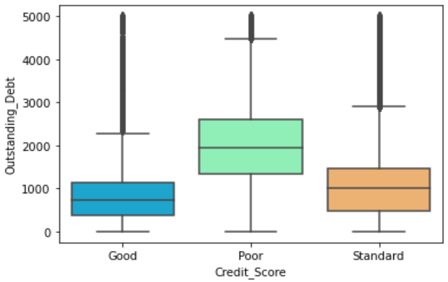












DATA / OUTLIERS TREATMENT

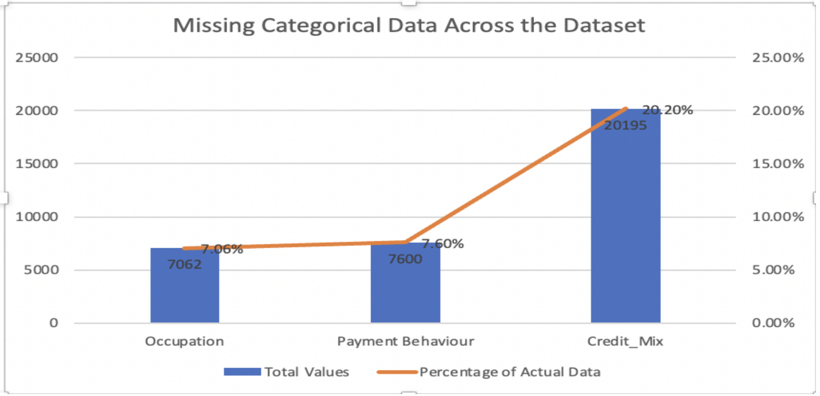
DATA TREATMENT | Replacing NULLs/ Missing Values in Categorical variables ; Converting Numerical columns to appropriate format, suitable for Models

```
credit_train_df['Credit_History_Age']

0          22 Years and 1 Months
1                            NaN
2          22 Years and 3 Months
3          22 Years and 4 Months
4          22 Years and 5 Months
                ...
99995      31 Years and 6 Months
99996      31 Years and 7 Months
99997      31 Years and 8 Months
99998      31 Years and 9 Months
99999     31 Years and 10 Months
```

```
credit_train_df['Credit_History_Age_YR']

0        22.083333
1              NaN
2        22.250000
3        22.333333
4        22.416667
           ...
99995    31.500000
99996    31.583333
99997    31.666667
99998    31.750000
99999    31.833333
```

Since the age is in Alpha Numeric format, we converted it into years by utilizing the below mentioned formula:

Age= Year part +(Month Part/12)



Replaced the Missing & In-appropriate Values in the categorical columns by "**Unknown**" to avoid causing Biasness in the data

Replaced the Missing & In-appropriate Values in the categorical columns by "**Unknown**" to avoid causing Biasness in the data

DATA TREATMENT | Replacing the NULLs in the Numerical Columns by the Median Value of each Occupation across the Dataset

```
#Checking the number of Nulls in dataframe
credit_train_df.isna().sum()
```

```
ID                            0
Customer_ID                   0
Month                         0
Name                       9985
Age                           0
SSN                           0
Occupation                    0
Annual_Income                 0
Monthly_Inhand_Salary     15002
Num_Bank_Accounts             0
Num_Credit_Card               0
Interest_Rate                 0
Num_of_Loan                   0
Type_of_Loan              11408
Delay_from_due_date           0
Num_of_Delayed_Payment     7002
Changed_Credit_Limit       2091
Num_Credit_Inquiries       1965
Credit_Mix                    0
Outstanding_Debt              0
Credit_Utilization_Ratio      0
Payment_of_Min_Amount         0
Total_EMI_per_month           0
Amount_invested_monthly    4479
Payment_Behaviour             0
Monthly_Balance            2868
Credit_Score                  0
Credit_History_Age_YR      9030
```

```
#Checking the percentage of Nulls/NaN/Na in dataset
credit_train_df.isna().sum()/len(credit_train_df)*100
```

```
ID                          0.000
Customer_ID                 0.000
Month                       0.000
Name                        9.985
Age                         0.000
SSN                         0.000
Occupation                  0.000
Annual_Income               0.000
Monthly_Inhand_Salary      15.002
Num_Bank_Accounts           0.000
Num_Credit_Card             0.000
Interest_Rate               0.000
Num_of_Loan                 0.000
Type_of_Loan               11.408
Delay_from_due_date         0.000
Num_of_Delayed_Payment      7.002
Changed_Credit_Limit        2.091
Num_Credit_Inquiries        1.965
Credit_Mix                  0.000
Outstanding_Debt            0.000
Credit_Utilization_Ratio    0.000
Payment_of_Min_Amount       0.000
Total_EMI_per_month         0.000
Amount_invested_monthly     4.479
Payment_Behaviour           0.000
Monthly_Balance             2.868
Credit_Score                0.000
Credit_History_Age_YR       9.030
dtype: float64
```

NULLs in Numerical columns contribute to **~40%** of the dataset since they don't occur together. Dropping them would lead to loss of fair number of records; making the dataset in sufficient for good prediction

We have replaced the Nulls by the Median of the data across each Occupation Sector since Occupation affects the Credit Score the most.

Sample of the Code for two Numerical Columns:

```
Median_by_group=credit_train_df.groupby(['Occupation'])['Monthly_Balance','Num_of_Delayed_Payment'].agg('median')
credit_train_df = credit_train_df.merge(Median_by_group,how='left', on='Occupation')
credit_train_df.loc[credit_train_df['Monthly_Balance_x'].isna(), 'Monthly_Balance_x'] = credit_train_df['Monthly_Balance_y']
credit_train_df.loc[credit_train_df['Num_of_Delayed_Payment_x'].isna(), 'Num_of_Delayed_Payment_x'] = credit_train_df['Num_of_Delayed_Payment_y']
```

```
#Checking the number of Nulls in dataframe
credit_train_df[['Monthly_Balance','Num_of_Delayed_Payment']].isna().sum()
```

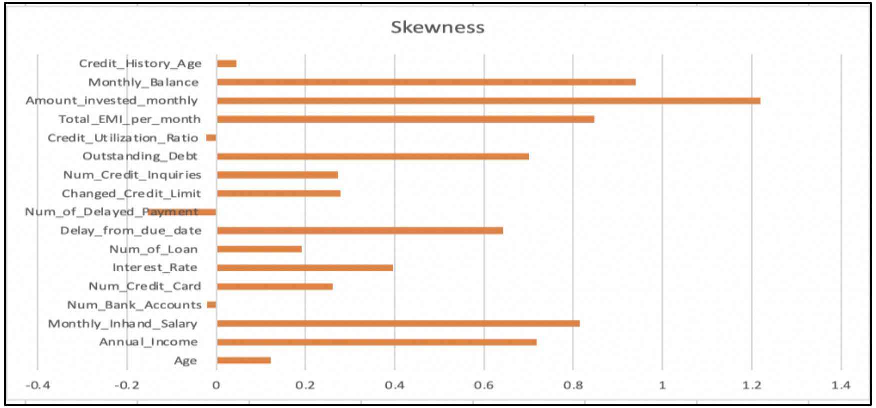
```
Monthly_Balance           0
Num_of_Delayed_Payment    0
dtype: int64
```

```
#Checking the percentage of Nulls/NaN/Na in dataset
credit_train_df[['Monthly_Balance','Num_of_Delayed_Payment']].isna().sum()/len(credit_train_df)*100
```

```
Monthly_Balance           0.0
Num_of_Delayed_Payment    0.0
dtype: float64
```

OUTLIERS TREATMENT | Using IQR and capping & flooring the outliers with 90th and 10th percentile values resulting in minimal loss and not biasing the dataset

```
#Doing outlier treatment by capping the upper limit to 90th percentile and flooring the lower limit to 10th percentile
for i in credit_train_df_not_object_columns:
    lower_bound=credit_train_df[i].quantile(0.10)
    upper_bound=credit_train_df[i].quantile(0.90)
    credit_train_df[i] = np.where(credit_train_df[i] <lower_bound, lower_bound,credit_train_df[i])
    credit_train_df[i] = np.where(credit_train_df[i] >upper_bound, upper_bound,credit_train_df[i])
```

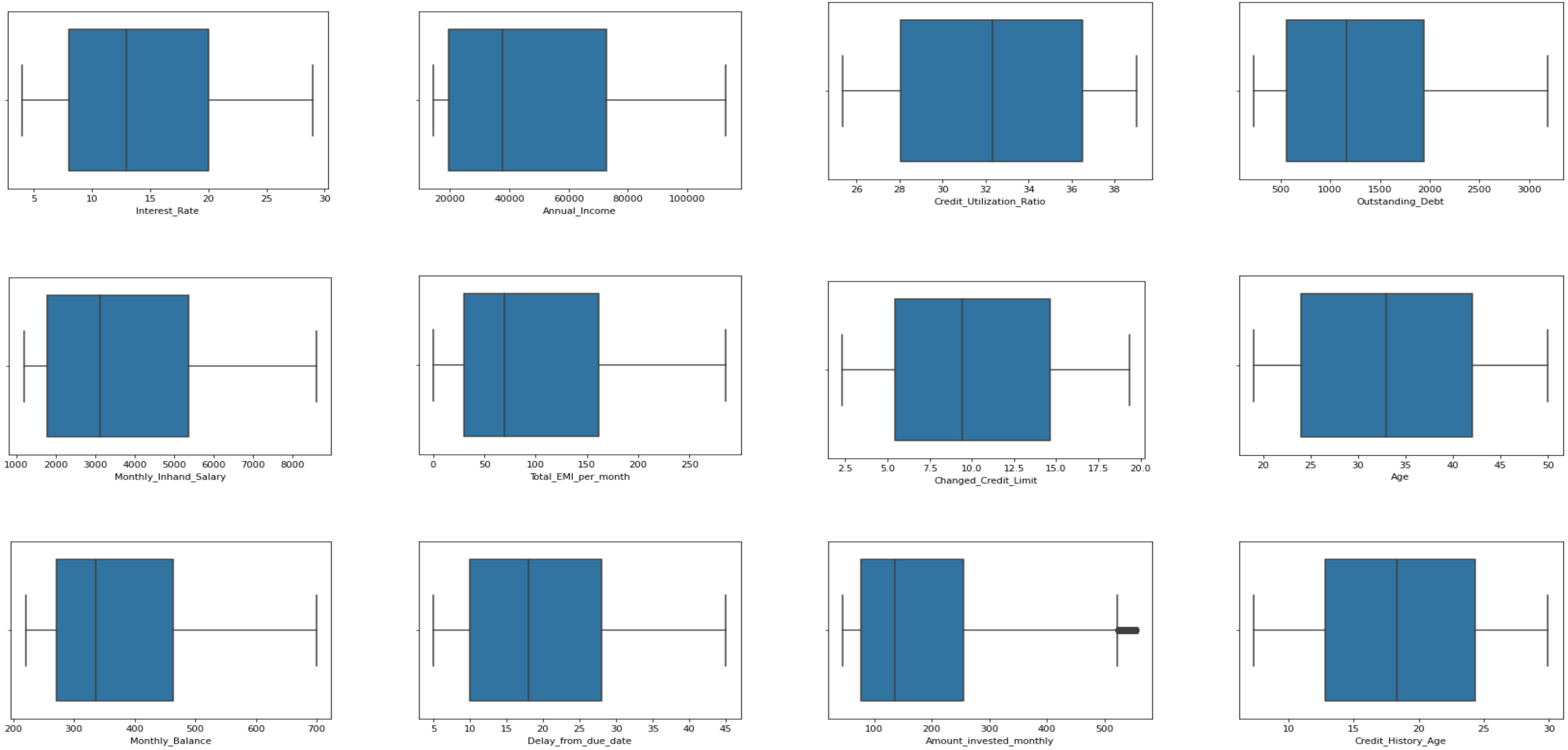


After performing Outlier Treatment, the skewness of the Features have reduced significantly

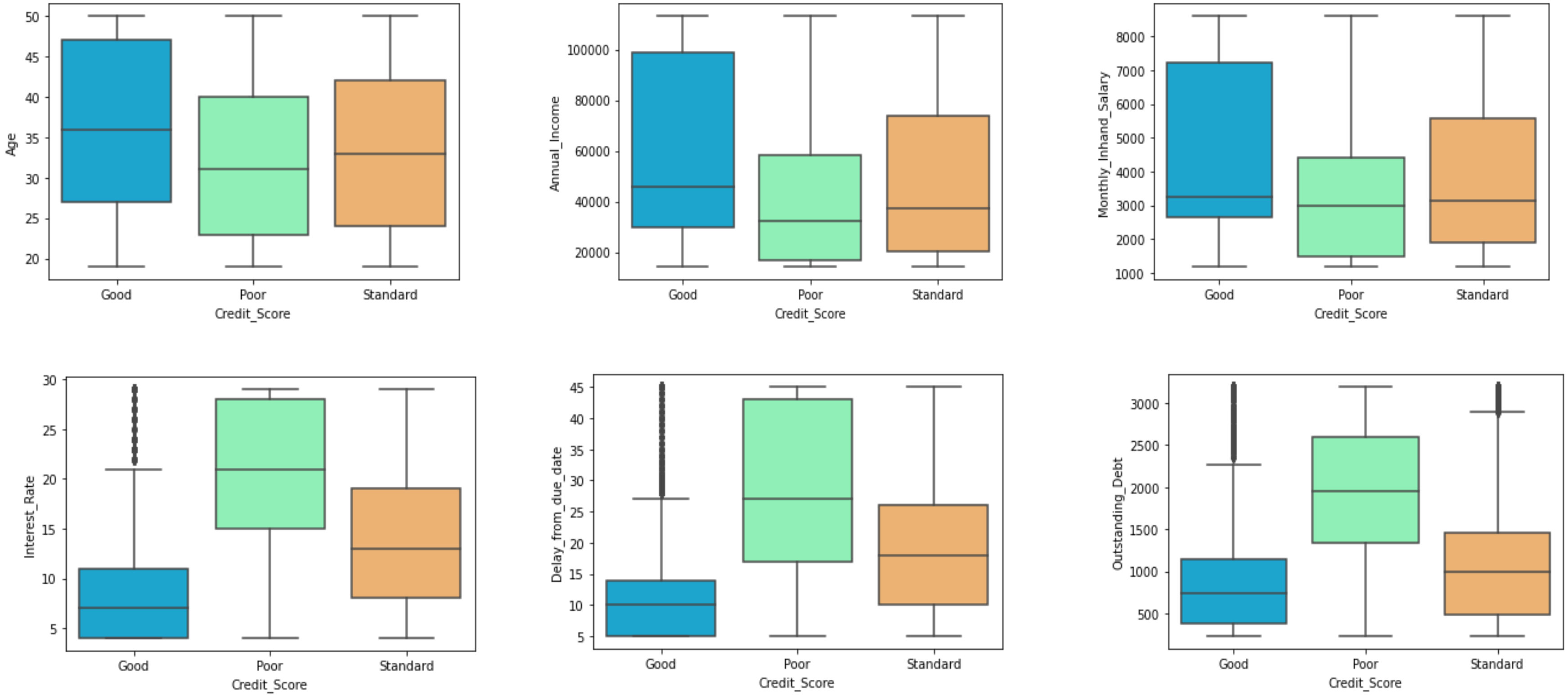
EXPLORATORY DATA ANALYSIS POST DATA TREATMENT

TEXAS
The University of Texas at Austin

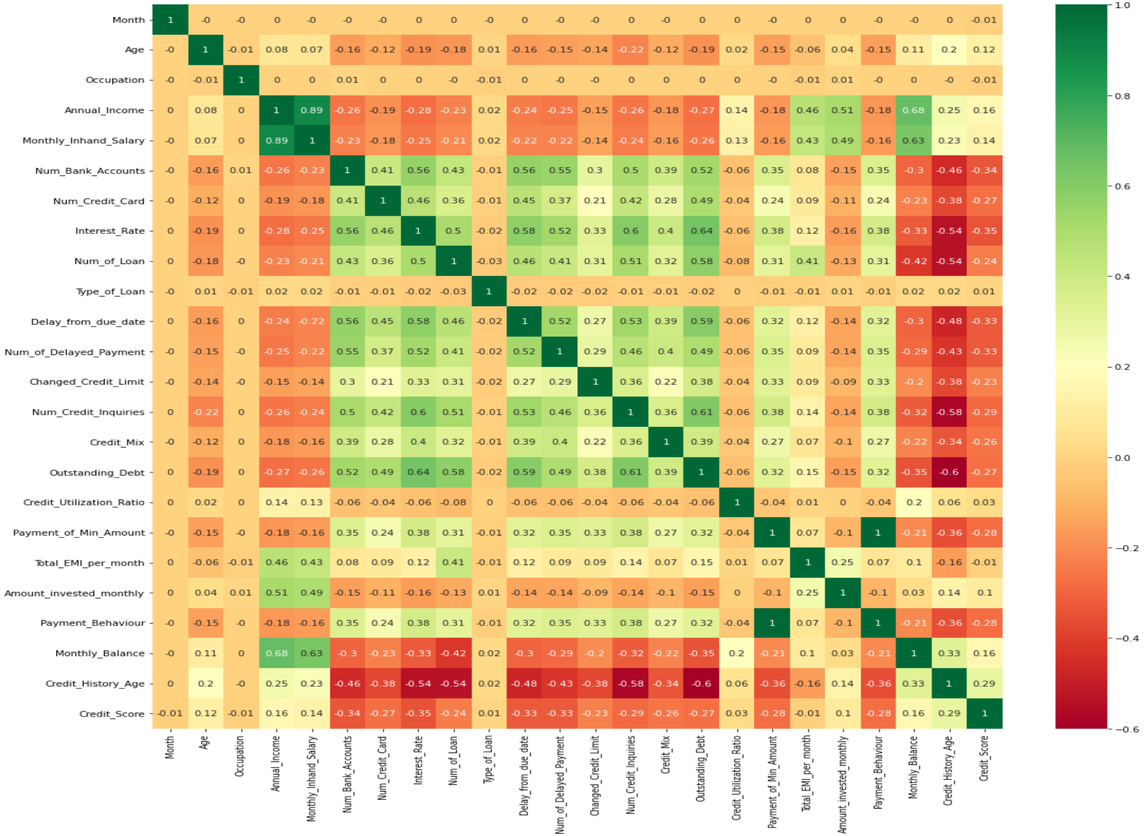
EDA | Uni-Variate Analysis | Boxplots for Numerical Variables to see the distribution after treating the Outliers



EDA | Bi-Variate Analysis | Box-Plots for Numerical Variables with respect to different Credit Score Categories



EDA | Bi-Variate Analysis | Heat Map for the correlation between the Credit Score and Numerical Columns



As observed, None of the Numerical features have Co-relation value of 0.8 or greater with the Target Variable

Note: Since Corr() function, requires the Target Variable to be Binary Class, we have encoded the Poor and Standard classes as 0 and Good as 1

MODELING

MODELING | Before feeding the training set into the model, we pre-processed the data, encoding the categorical columns into multiclass format

- We have used two different encoding methods for categorical columns
  - Label Encoding:

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
for col in credit_train_df.select_dtypes(include='category').columns:
    if col=='Credit_Score':
        pass
    else:
        credit_train_df[col]=le.fit_transform(credit_train_df[col])
```

  - Ordinal Encoding

```
def ordinal_encoder(data,feature,feature_rank):

    ordinal_dict = {}

    for i, feature_value in enumerate(feature_rank):
        ordinal_dict[feature_value]=i+1

    data[feature] = data[feature].map(lambda x: ordinal_dict[x])

    return data
```

```
ordinal_encoder(credit_train_df,'Credit_Score',['Good', 'Standard', 'Poor']).head()
```

- We have used MinMaxScaler() to normalize the data: the min-max scaling has been applied to all feature columns

```
# Feature Scaling for input features.
scaler = preprocessing.MinMaxScaler()
x_scaled = scaler.fit_transform(credit_X)
test_x_scaled = scaler.fit_transform(credit_Test_X)
```

- We have split the data into 3 set: Training, Validation and Testing

MODELING | After doing data cleaning and preprocessing, we are building classification models to predict the credit score category

- We will try few models as below:
  - **Logistic Regression**
  - **KNN**
  - **Decision Trees**
  - **Random Forest Classifier**
  - **Gradient Boosting**
- We will use stratified K-Fold cross-validation to evaluate our models

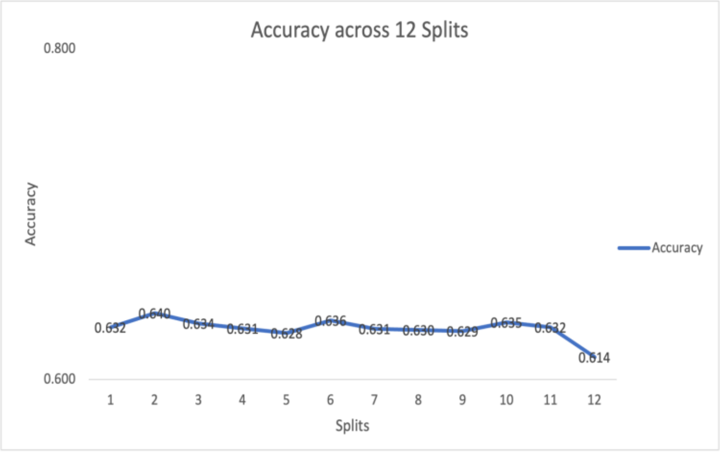
MODELING | LOGISTIC REGRESSION WITH STRATIFIED K-FOLD CROSS VALIDATION

feature	importance
Delay_from_due_date	-1.291446
Interest_Rate	-1.084787
Num_Credit_Card	-0.834258
Credit_Mix	-0.642493
Payment_of_Min_Amount	-0.538083
Num_of_Loan	-0.465725
Outstanding_Debt	-0.373964
Num_Credit_Inquiries	-0.340312
Num_Bank_Accounts	-0.264900
Monthly_Balance	-0.250082
Num_of_Delayed_Payment	-0.154494
Occupation	-0.073392
Month	-0.053501
Amount_invested_monthly	-0.042517
Changed_Credit_Limit	-0.028272
Credit_Utilization_Ratio	-0.027055
Type_of_Loan	-0.002307
Annual_Income	0.099172
Age	0.107044
Credit_History_Age	0.428965
Total_EMI_per_month	0.482900

```
Results based on Train-Validation Set:


Maximum Accuracy that can be obtained from this model is: 64.01679160042 %
Minimum Accuracy: 61.35613561356136 %
Overall Accuracy: 63.088732002043756 %
Standard Deviation is: 0.006358654350523413
```

```
Running the Model on Test Data


Confusion Matrix:
[[1670 2011   57]
[1014 8134 1547]
[ 222 2346 2999]]

Classification Report :
              precision    recall  f1-score   support

           1       0.57      0.45      0.50      3738
           2       0.65      0.76      0.70     10695
           3       0.65      0.54      0.59      5567

    accuracy                           0.64     20000
   macro avg       0.63      0.58      0.60     20000
weighted avg       0.64      0.64      0.63     20000


Accuracy for Test Data Set 64.015

Recall for Test Data Set 64.015

Precision for Test Data Set 63.69834648772217

F1_Score for Test Data Set 63.331670104425356
```

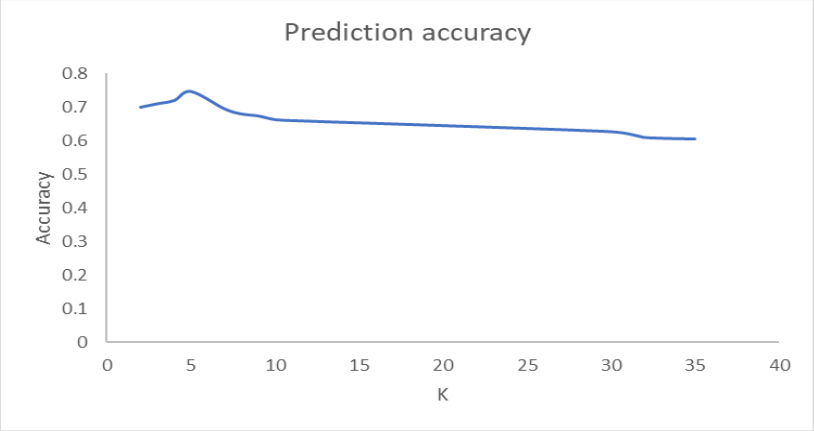
For cross validation, we have assigned number of splits=12



MODELING | K-NEAREST NEIGHBOURS WITH STRATIFIED K-FOLD CROSS VALIDATION

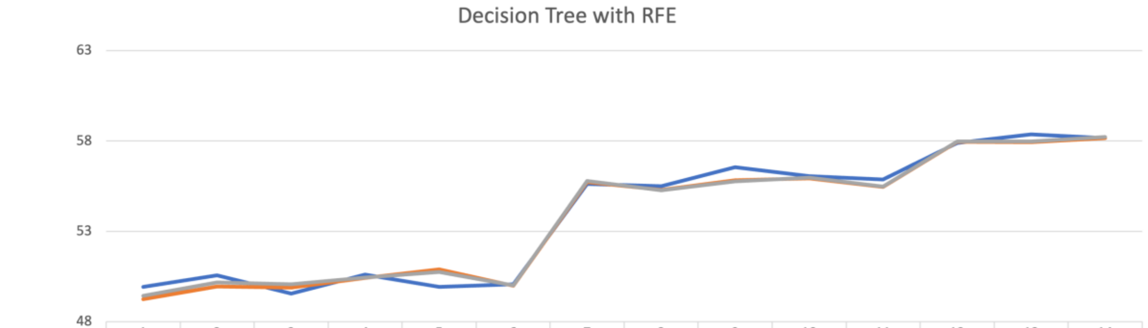
```
Running the Model on Test Data


Confusion Matrix:
[[1480   65  675]
 [ 109 2909  659]
 [ 713  938 4952]]

Classification Report :
              precision    recall  f1-score   support

           0       0.64      0.67      0.65      2220
           1       0.74      0.79      0.77      3677
           2       0.79      0.75      0.77      6603

    accuracy                           0.75     12500
   macro avg       0.72      0.74      0.73     12500
weighted avg       0.75      0.75      0.75     12500


Accuracy for Test Data Set: 74.73 %
Precision for Test Data Set: 75 %
Recall for Test Data Set: 75 %
F1_Score for Test Data Set: 75 %
```

KNN: Average out the nearest k-neighbors to predict the value
- Maximum accuracy of 74.73% was obtained for K nearest neighbors = 5



MODELING | DECISION TREE WITH RECURSIVE FEATURE ELIMINATION

```
Number of Features giving highest accuracy for Decision Tree  : 13
Maximum Accuracy for Decision Tree with Recursive Feature Elimination is  : 58.365
```



	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Accuracy	49.925	50.575	49.55	50.61	49.92	50.08	55.62	55.49	56.555	56.055	55.865	57.9	58.365	58.15
Precision	49.25	49.95	49.89	50.43	50.89	49.98	55.72	55.29	55.82	55.92	55.46	57.96	57.94	58.15
F1_Score	49.43	50.18	50.06	50.44	50.74	50.01	55.79	55.28	55.77	55.97	55.47	57.96	57.98	58.22

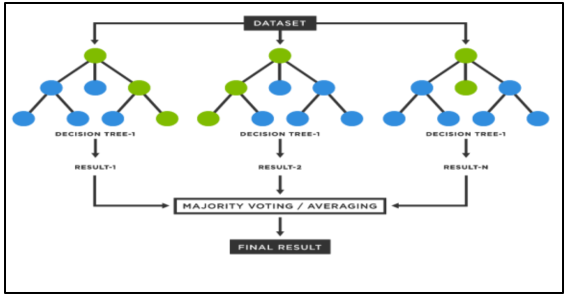
Number of Features selected in RFE

Accuracy Precision F1_Score

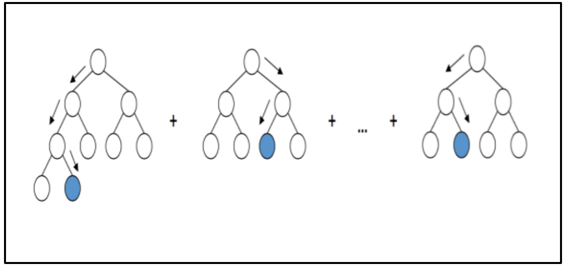
MODELING | Ensemble Learning Techniques

An approach that seeks better predictive performance by combining the predictions from multiple models

- **Random Forest**



- **Gradient Boosting**

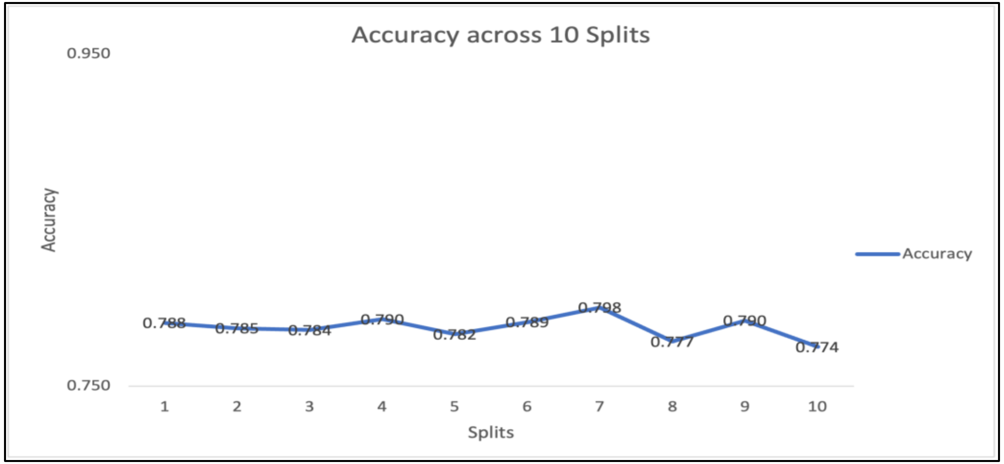


MODELING | RANDOM FOREST CLASSIFIER WITH STRATIFIED K-FOLD CROSS VALIDATION

```
Results based on Train-Validation Set:

List of possible accuracy based on Train-Validation Set: [0.78825, 0.785, 0.784, 0.790375, 0.7815, 0.78875, 0.7975, 0.777125, 0.789625, 0.773875]

Maximum Accuracy that can be obtained from this model is: 79.75 %
Minimum Accuracy: 77.3875 %
Overall Accuracy: 78.56 %
Standard Deviation is: 0.006884664923662681
```

```
Running the Model on Test Data


Confusion Matrix:
[[2420 1243   75]
 [1379 7975 1341]
 [ 339 1551 3677]]

Classification Report :
              precision    recall  f1-score   support

           1       0.58      0.65      0.61      3738
           2       0.74      0.75      0.74     10695
           3       0.72      0.66      0.69      5567

    accuracy                           0.70     20000
   macro avg       0.68      0.68      0.68     20000
weighted avg       0.71      0.70      0.70     20000


Accuracy for Test Data Set 70.36

Recall for Test Data Set 70.36

Precision for Test Data Set 70.6274208013408

F1_Score for Test Data Set 70.42549485611853
```

For cross validation, we have assigned number of splits=10

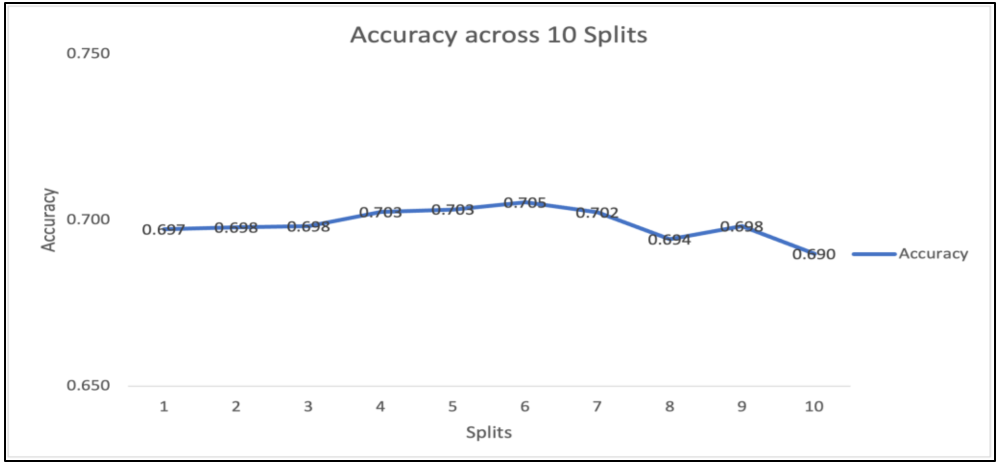


MODELING | GRADIENT BOOSTING WITH STRATIFIED K-FOLD CROSS VALIDATION

```
Results based on Train-Validation Set:

List of possible accuracy based on Train-Validation Set: [0.69725, 0.697875, 0.69825, 0.7025, 0.70325, 0.705375, 0.702375, 0.69425, 0.69825, 0.689875]

Maximum Accuracy that can be obtained from this model is: 70.5375 %
Minimum Accuracy: 68.9875 %
Overall Accuracy: 69.8925 %
Standard Deviation is: 0.004637632897167345
```

```
Running the Model on Test Data


Confusion Matrix:
[[2339 1324   75]
 [1342 8181 1172]
 [ 294 1682 3591]]

Classification Report :
              precision    recall  f1-score   support

           1       0.59      0.63      0.61      3738
           2       0.73      0.76      0.75     10695
           3       0.74      0.65      0.69      5567

    accuracy                           0.71     20000
   macro avg       0.69      0.68      0.68     20000
weighted avg       0.71      0.71      0.71     20000


Accuracy for Test Data Set 70.555

Recall for Test Data Set 70.555

Precision for Test Data Set 70.76422399037833

F1_Score for Test Data Set 70.5338977905536
```

For cross validation, we have assigned number of splits=10



MODEL COMPARISON

MODEL COMPARISON | MODEL ACCURACY COMPARISONS

Classifier	Train + Validation	Test
Logistic Regression	64.0%	64.0%
KNN	83.6%	74.7%
Decision Tree (RFE)	62.3%	58.4%
Gradient Boosting	70.5%	70.5%
Random Forest	79.8%	70.4%

THANK YOU