LOAN PREDICTION

GROUP-1

Mentor Name: Mr. Parth

Date: 21-05-2021

EXCELR
Raising Excellence

Prepared by: Amit Kulkarni

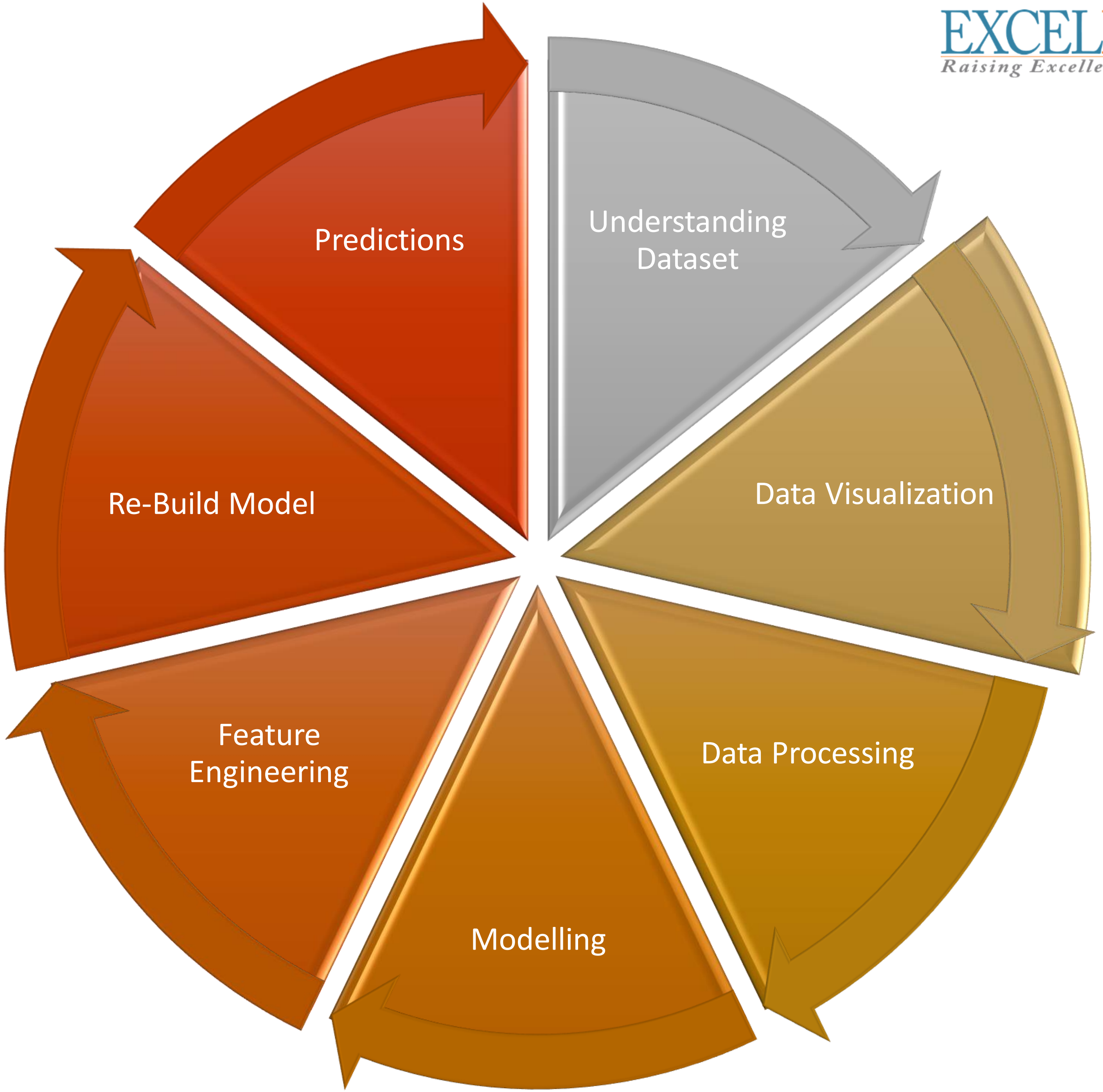
Business Problem:

To predict the impact of the incident raised by the customer.

Objective:

To automate the loan eligibility process (real time) based on customer detail provided while filling online application form.

PROJECT FLOW



Understanding Data-Set

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            614 non-null    object
 1   Gender             601 non-null    object
 2   Married            611 non-null    object
 3   Dependents         599 non-null    object
 4   Education          614 non-null    object
 5   Self_Employed      582 non-null    object
 6   ApplicantIncome    614 non-null    int64
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object
 12  Loan_Status        614 non-null    object
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
```

```
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 367 entries, 0 to 366
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            367 non-null    object
 1   Gender             356 non-null    object
 2   Married            367 non-null    object
 3   Dependents         357 non-null    object
 4   Education          367 non-null    object
 5   Self_Employed      344 non-null    object
 6   ApplicantIncome    367 non-null    int64
 7   CoapplicantIncome  367 non-null    int64
 8   LoanAmount         362 non-null    float64
 9   Loan_Amount_Term   361 non-null    float64
 10  Credit_History     338 non-null    float64
 11  Property_Area      367 non-null    object
dtypes: float64(3), int64(2), object(7)
memory usage: 34.5+ KB
```

Understanding Data-Set

```
train.isnull().sum().sort_values(ascending=False)
```

```
Credit_History       50
Self_Employed        32
LoanAmount           22
Dependents           15
Loan_Amount_Term     14
Gender               13
Married               3
Loan_Status           0
Property_Area         0
CoapplicantIncome     0
ApplicantIncome       0
Education             0
Loan_ID               0
dtype: int64
```

```
test.isnull().sum().sort_values(ascending=False)
```

```
Credit_History       29
Self_Employed        23
Gender               11
Dependents           10
Loan_Amount_Term      6
LoanAmount            5
Property_Area         0
CoapplicantIncome     0
ApplicantIncome       0
Education             0
Married               0
Loan_ID               0
dtype: int64
```

Inferences from given Data-Set

- There are **614 records in the Train Dataset & 367 records in the Test Dataset.**
- There are **no duplicated data** in Train set.

1. Training Dataset has:

- Loan_ID | Gender | Married | Dependents | Education | Self_Employed | Property_Area | Loan_status as **Object types.**
- ApplicantIncome field is of **Integer type.**
- The other 3 fields namely CoapplicantIncome | Loan_Amount_Term | Credit_History are **Floating point type.**

2. Testing Dataset has:

- CoapplicantIncome is of **Integer Type not Floating**
- There is no column as ***Loan_status,*** that's what we have to ***predict by creating a model.***

Inferences from given Data-Set

3. Null Values in Train Data :

- 50 | Credit_History
- 32 | Self_Employed
- 22 | LoanAmount
- 15 | Dependents
- 14 | Loan_Amount_Term
- 13 | Gender
- 03 | Married

4. Null Values in Test Data :

- 29 | Credit_History
- 23 | Self_Employed
- 11 | Gender
- 10 | Dependents
- 06 | Loan_Amount_Term
- 05 | LoanAmount

Null values would be Treated post Analysis

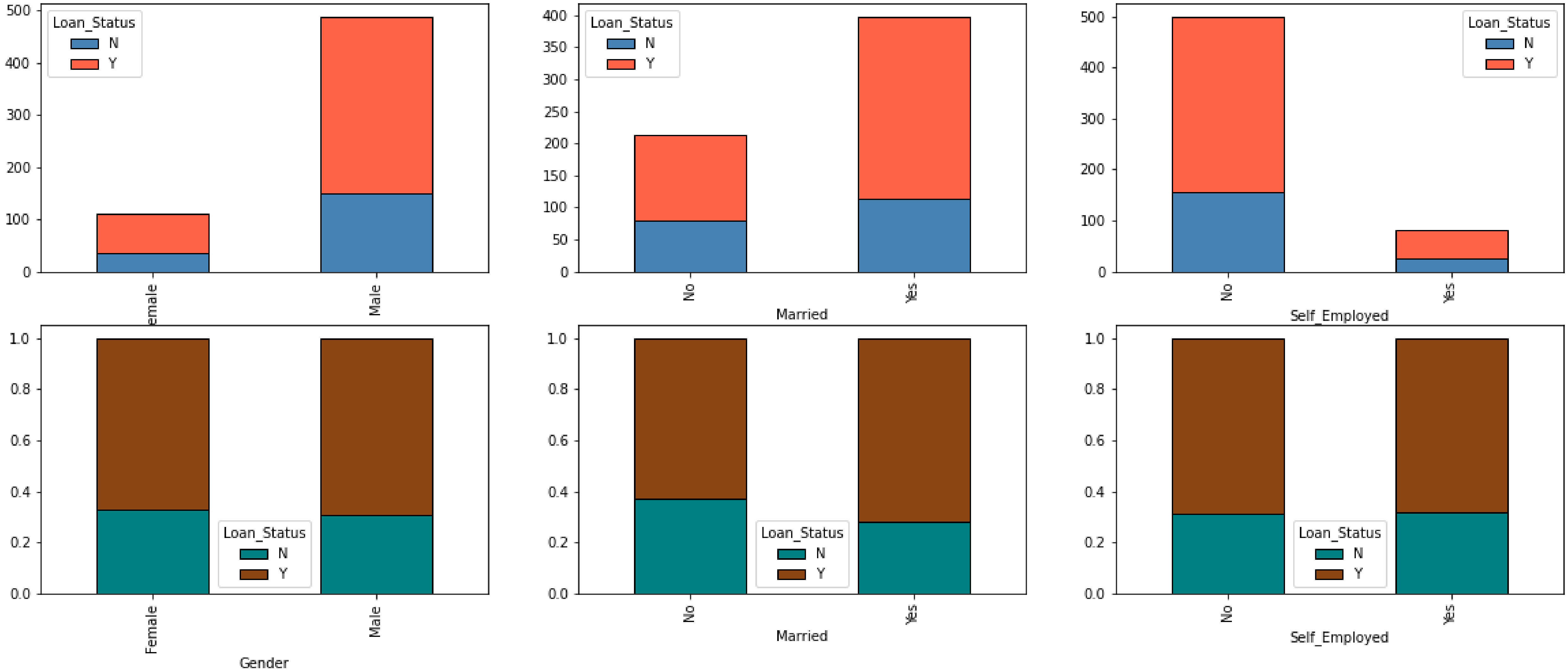
Hypothesis Generation :

- Salary: Applicants with **higher income** should **have higher chances** of loan approval
- Credit history: Applicants who have **followed Credit guidelines** should **have better chances**
- Loan Amount: **Lower Loan Amounts** should **have better chances** of approval
- Loan Amount Terms: **Shorter tenures** should **have better chances** of approval
- EMI: **Lower the expected monthly installment** for the applicant, compared to income, the **better** the approval chances

EDA

Univariate Analysis on Categorical Data

EXCELR
Raising Excellence

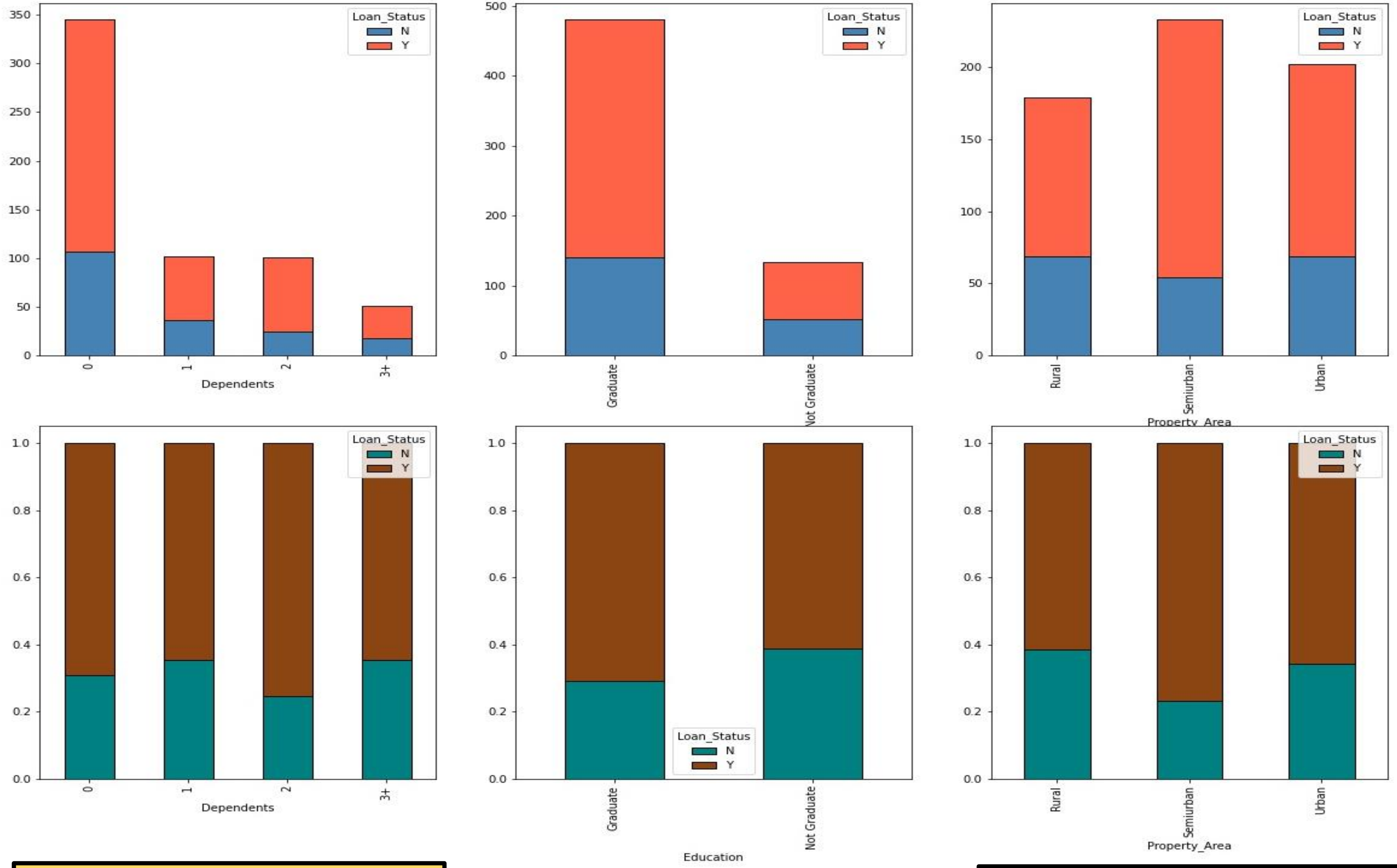


80% are Male Applicants, however both Male and Female have got equal proportion of Loan Approval

65% are Married and those who are Married have more chances to get Loan Approved

85% are Self_Employed, however equal proportion of Loan Approval

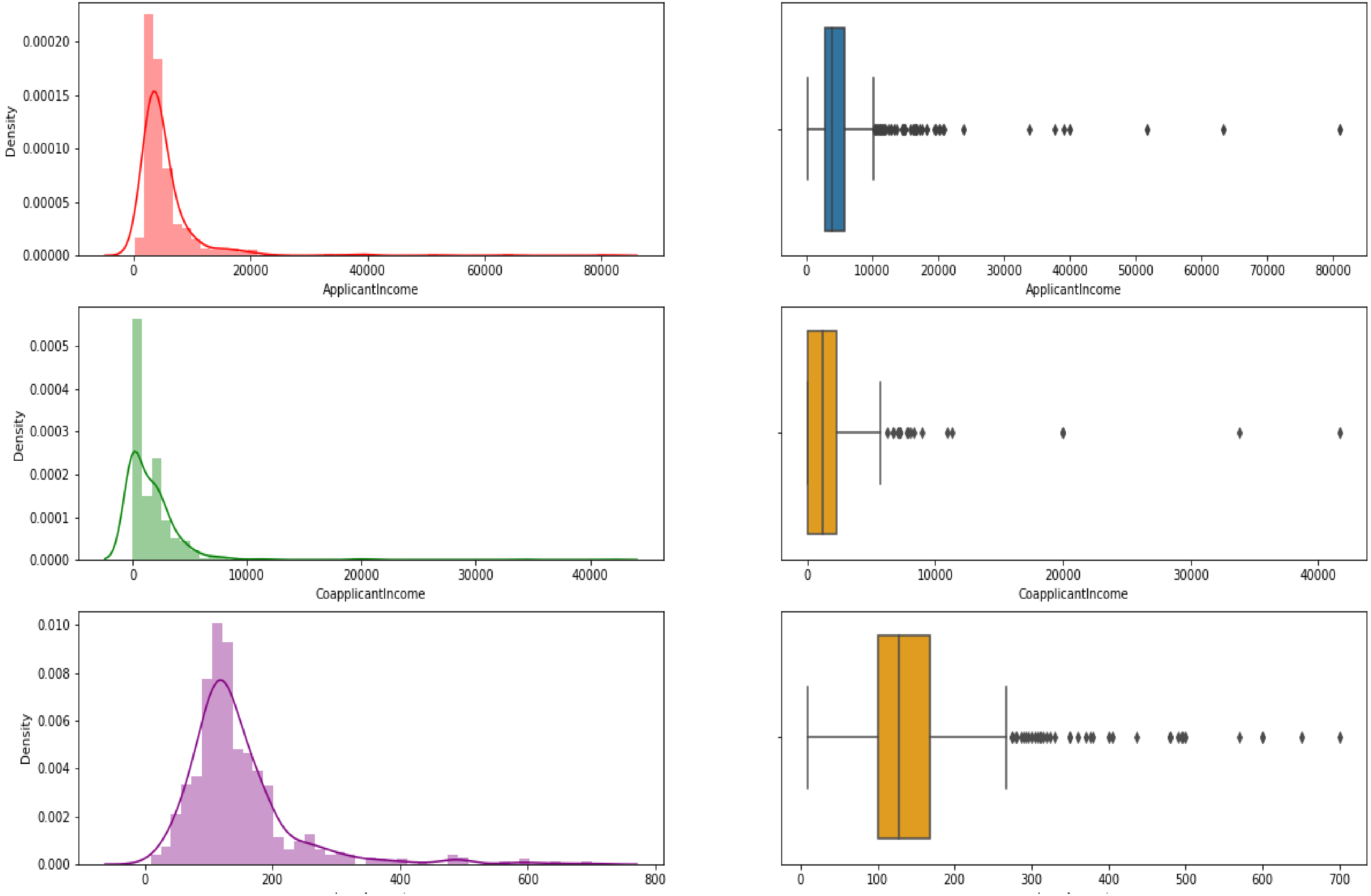
Univariate Analysis on Ordinal Data



- •57% are Parents have no Dependents (Newly Married or No Child).
- •However Bank has preferred to give Loan to Couple having 2 Dependents

- •78% are Graduate.
- •Those who are Graduate have greater chances of Loan Approval

- •37% are having Property in Semi-Urban area.
- • Those having property in Semi-Urban area have greater chances of Loan Approval

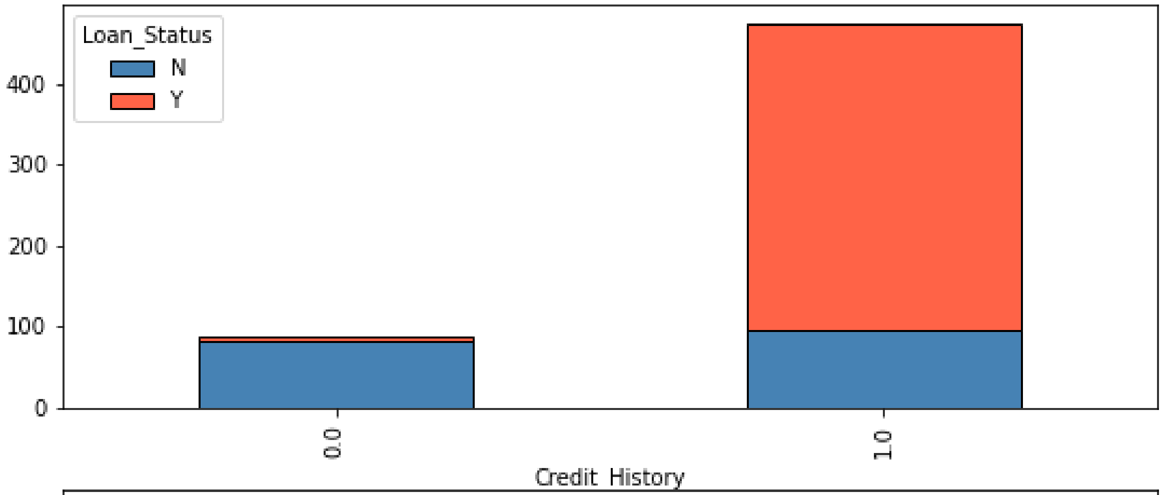


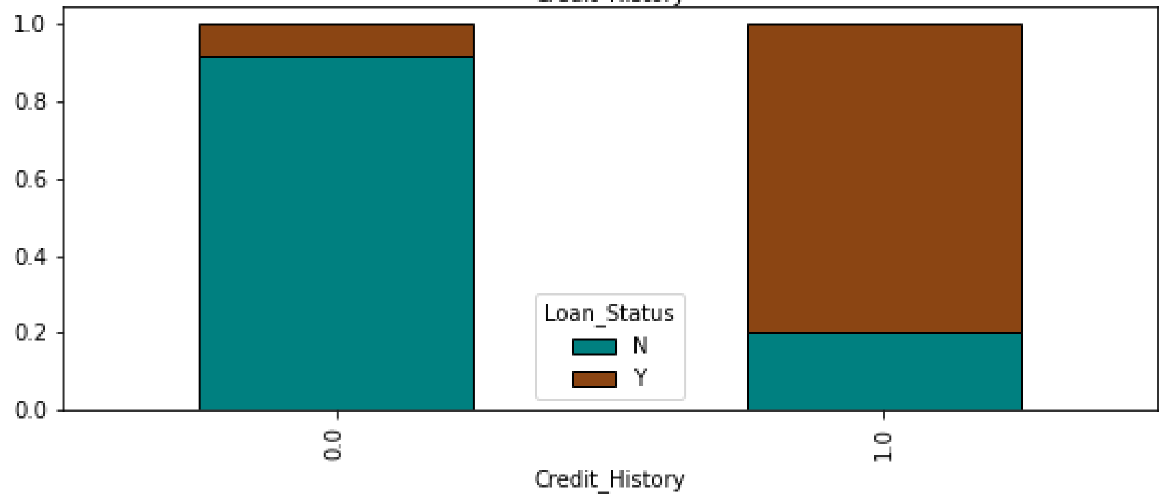
Univariate Analysis on Numerical Data

Most of the data seems to be right skewed as data congestion towards left side.

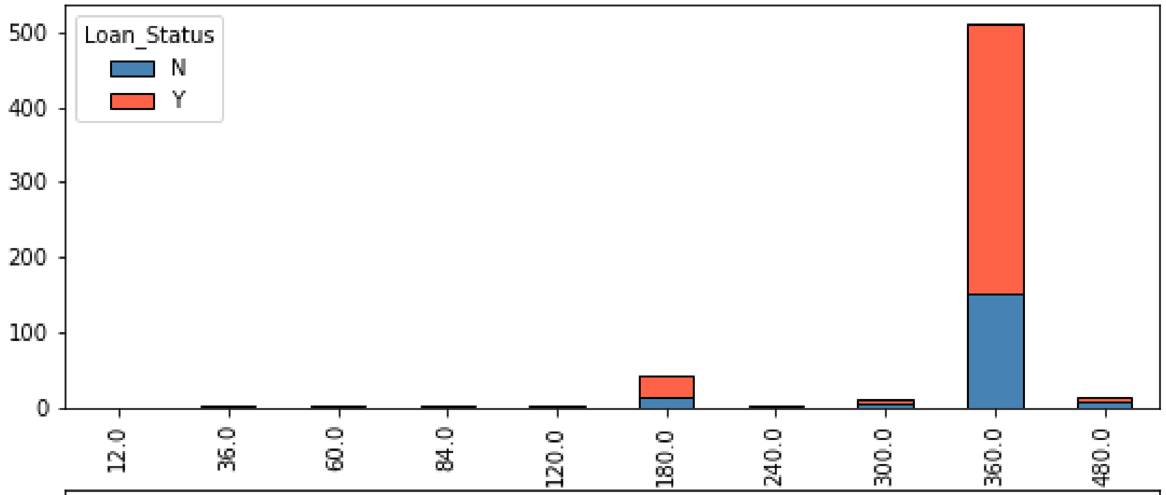
Also from the data we can conclude that if mean>median (Right Skew) else vice versa.

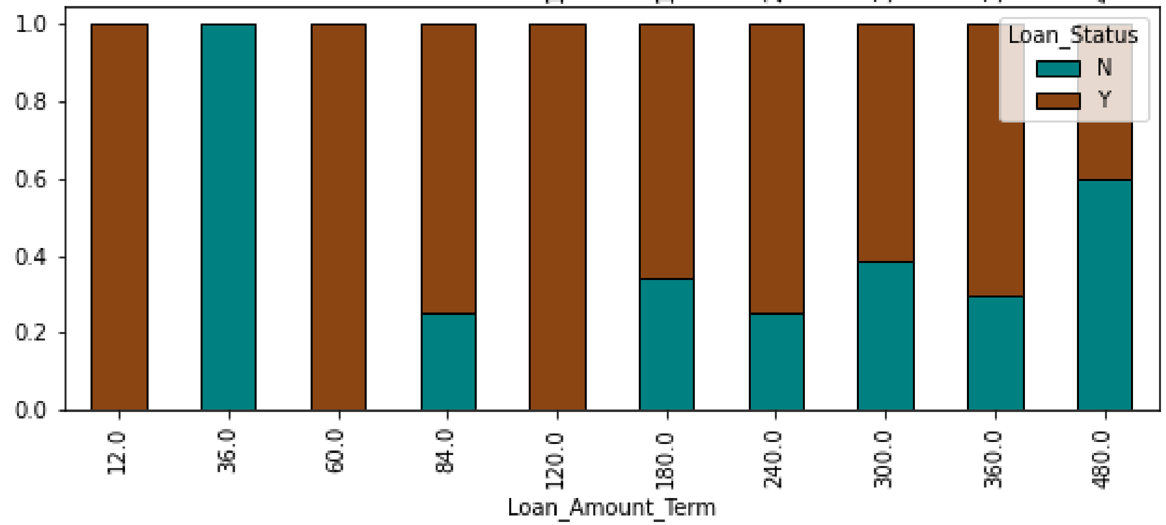
Univariate Analysis on Numerical Data





84% have Credit History as per guidelines, and mostly those whose Credit History meets guidelines have got Loan Approval





85% have 360 Months (30 Years) of Loan_Amount_Term, however people having Loan_Amount_Term of 1,5, & 10 Year have maximum chances of Loan Approval

Inferences from Data Visualization

1.1 Independent Variable [Categorical Type] :	Gender & Self_Employed Status doesn't make significant difference in getting Loan.
	If you are Married then more chances of getting Loan.
1.2 Indpendent Variable [Ordinal Type] :	Parents with 2 Dependents have more chances of approval.
	If you are graduated than more chances of Loan approval.
	People owning property in Semi-Urban area have greater chances of Loan Approval
1.3 Indpendent Variable [Numerical Type] :	Most of data is Right Skewed except Loan_Amount_Term
	Data also has some outliers present, thereby needs Data Transformation
	People having Credit History as per guidelines and Loan_Amount_Term of 1,5, & 10 Years have got Loan approval
1.4 Dependent Variable [Target]	Almost 70% have Loan Approved

Hypothesis Post Data Analysis

Observations

Gender and Self_Employed have no significant effect on Loan_Status

If an applicant is **Graduated** having property in **Semi-Urban** area and is **Married** with 2 **Dependents** and Credit History as per **Guidelines** with Loan_Amount_Term of **1,5,10 Year** then **Loan is Approved**

Hypothesis

It is not necessary that higher income means higher loan amount for both Applicant Income and Coapplicant Income

Gender & Self_Employed is not a criteria for considering Loan Approval

Higher Loan Amount Lesser the chance of getting Loan Approved

Data Processing

Treating Null Values

- Mode Imputation on Gender and Married Column

Mode Imputation on Married | Gender

```
# Married Imputed with Mode
train['Married'].fillna(train['Married'].mode()[0],inplace=True)

# Gender Imputed with Mode
train['Gender'].fillna(train['Gender'].value_counts().index[0], inplace=True)
test['Gender'].fillna(test['Gender'].value_counts().index[0], inplace=True)
```

- Conditional Imputation on Dependents | Credit History | Self Employed | Loan Amount Term | Loan Amount
- Mapping Property Area

Mapping Property_Area

```
train["Property_Area"] = train["Property_Area"].map({"Rural":0, "Semiurban":1, "Urban": 2,})
test["Property_Area"] = test["Property_Area"].map({"Rural":0, "Semiurban":1, "Urban": 2,})
```

Treating Null Values

- Conditional Imputation on Dependents | Credit History | Self Employed

Conditional Class Imputation on Dependents | Credit_History | Self_Employed

```
# Replace Null value in Dependents with 2 if Loan is Approved otherwise Dependents = 1 if Loan is Not Approved
train.loc[(train.Dependents.isnull())&(train.Loan_Status==1),'Dependents'] = '2'
train.loc[(train.Dependents.isnull()),'Dependents'] = '1'

test.loc[(test.Dependents.isnull())&(test.Credit_History==1),'Dependents'] = '2'
test.loc[(test.Dependents.isnull()),'Dependents'] = '1'

train["Dependents"] = train["Dependents"].map({"0": 0, "1": 1,"2": 2, "3+": 3})
test["Dependents"] = test["Dependents"].map({"0": 0, "1": 1,"2": 2, "3+": 3})

# Credit_History = 1 have majority Loan Approved so we impute it inplace of NaN values otherwise Credit_History = 0
train.loc[(train.Credit_History.isnull())&(train.Loan_Status==1),'Credit_History'] = 1
train.loc[(train.Credit_History.isnull()),'Credit_History'] = 0

test['Credit_History'].fillna(test['Credit_History'].value_counts().index[0], inplace=True)
#In test data, for the user with income = 2733.
# To impute credit history as 0 based upon the Income to loan ratio
test.loc[(test.ApplicantIncome == 2733),'Credit_History']  = 0

# Self-Employed = No if Credit History is 1 Else Yes
train.loc[(train.Self_Employed.isnull())&(train.Credit_History==1),'Self_Employed'] ='No'
train.loc[(train.Self_Employed.isnull()),'Self_Employed'] = 'Yes'

test.loc[(test.Self_Employed.isnull())&(test.Credit_History==1),'Self_Employed'] ='No'
test.loc[(test.Self_Employed.isnull()),'Self_Employed'] = 'Yes'
```

Treating Null Values

- Conditional Imputation on Loan Amount

```
#  LoanAmount depends upon Property_Area, married, education, self employed and dependent columns.
# Hence we will group them by above features and input median values.
# In case if the median is null then we will impute median of the entire LoanAmount column.
LoanAmount_train_nan = list(train["LoanAmount"][train["LoanAmount"].isnull()].index)

for i in LoanAmount_train_nan :
    LoanAmount_train_med = train["LoanAmount"].median() # find median of entire LoanAmount column
    LoanAmount_train_impute = train["LoanAmount"][((train['Property_Area'] == train.iloc[i]["Property_Area"]) & (train['Married'] == train.iloc[i]["Marr
    if not np.isnan(LoanAmount_train_impute) :
        train['LoanAmount'].iloc[i] = LoanAmount_train_impute
    else :
        train['LoanAmount'].iloc[i] = LoanAmount_train_med
```

```
LoanAmount_test_nan = list(test["LoanAmount"][test["LoanAmount"].isnull()].index)

for i in LoanAmount_test_nan :
    LoanAmount_test_med = test["LoanAmount"].median()
    LoanAmount_test_impute = test["LoanAmount"][((test['Property_Area'] == test.iloc[i]["Property_Area"]) & (test['Married'] == test.iloc[i]["Married"])
    if not np.isnan(LoanAmount_test_impute) :
        test['LoanAmount'].iloc[i] = LoanAmount_test_impute
    else :
        test['LoanAmount'].iloc[i] = LoanAmount_test_med
```

Treating Null Values

- Conditional Imputation on Loan Amount Term

```
# Loan_Amount_Term. Loan_Amount_Term depends upon married, education, self employed and dependent columns.
# Hence we will group them by above features and impute median values.
# In case if the median is null then we will impute median of the entire Loan_Amount_Term column
Loan_Amount_Term_train_nan = list(train["Loan_Amount_Term"][train["Loan_Amount_Term"].isnull()].index)

for i in Loan_Amount_Term_train_nan :
    Loan_Amount_Term_train_med = train["Loan_Amount_Term"].median() # find median of entire Loan_Amount_Term column
    Loan_Amount_Term_train_impute = train["Loan_Amount_Term"][(((train['Married'] == train.iloc[i]["Married"]) & (train['Education'] == train.iloc[i]["E
    if not np.isnan(Loan_Amount_Term_train_impute) :
        train['Loan_Amount_Term'].iloc[i] = Loan_Amount_Term_train_impute
    else :
        train['Loan_Amount_Term'].iloc[i] = Loan_Amount_Term_train_med
```

```
Loan_Amount_Term_test_nan = list(test["Loan_Amount_Term"][test["Loan_Amount_Term"].isnull()].index)

for i in Loan_Amount_Term_test_nan :
    Loan_Amount_Term_test_med = test["Loan_Amount_Term"].median()
    Loan_Amount_Term_test_impute = test["Loan_Amount_Term"][(((test['Married'] == test.iloc[i]["Married"]) & (test['Education'] == test.iloc[i]["Educati
    if not np.isnan(Loan_Amount_Term_test_impute) :
        test['Loan_Amount_Term'].iloc[i] = Loan_Amount_Term_test_impute
    else :
        test['Loan_Amount_Term'].iloc[i] = Loan_Amount_Term_test_med
```

Treating Null Values

```
train.isnull().sum(),test.isnull().sum()
```

```
(Loan_ID              0
 Gender               0
 Married              0
 Dependents           0
 Education            0
 Self_Employed        0
 ApplicantIncome      0
 CoapplicantIncome    0
 LoanAmount           0
 Loan_Amount_Term     0
 Credit_History       0
 Property_Area        0
 Loan_Status          0
 dtype: int64,
```

Train

```
Loan_ID              0
Gender               0
Married              0
Dependents           0
Education            0
Self_Employed        0
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           0
Loan_Amount_Term     0
Credit_History       0
Property_Area        0
dtype: int64)
```

Test

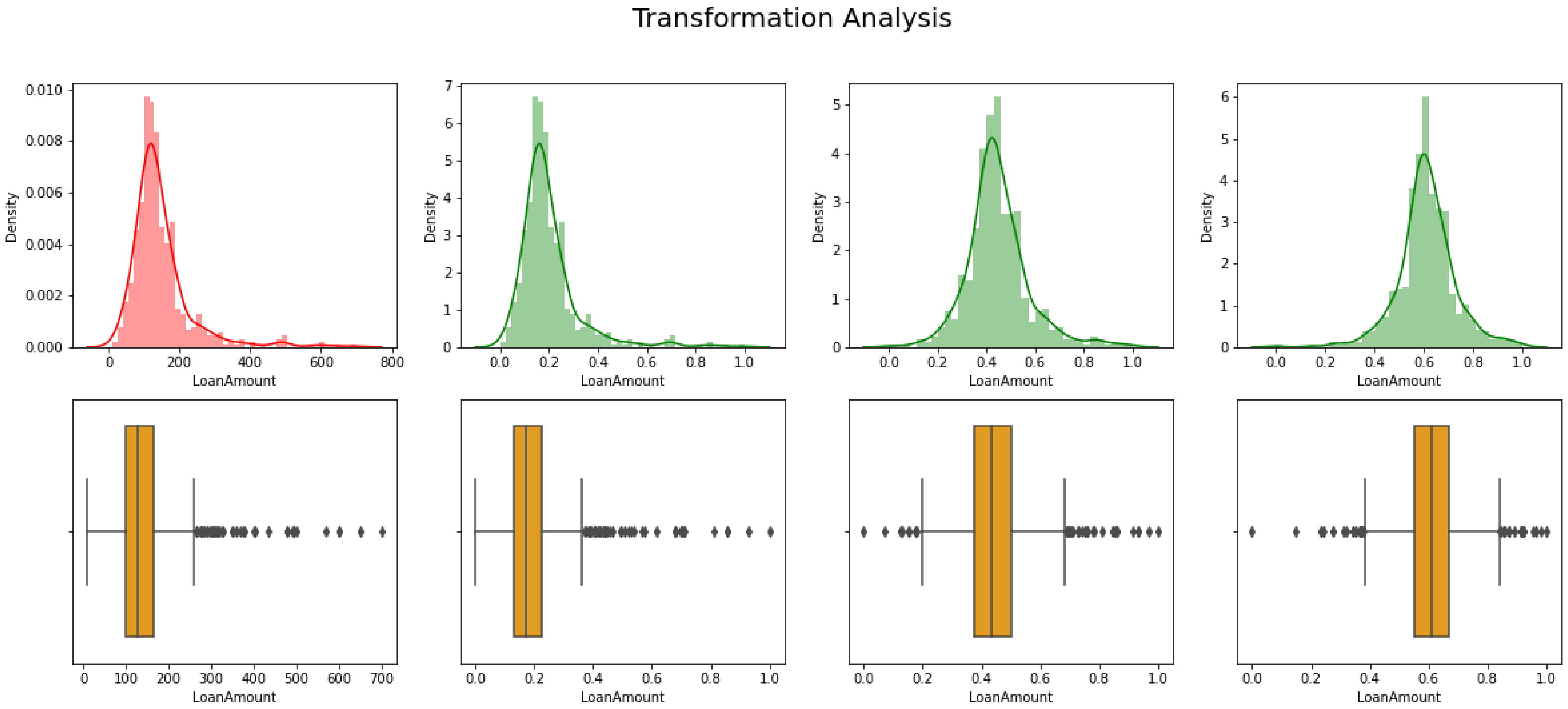
Treating Outliers

Treating Outliers

```
def norm_func(i):
    x = (i-i.min())/(i.max()-i.min())
    return (x)
```

```
import warnings
warnings.filterwarnings('ignore')
fig, axes = plt.subplots(2,4, figsize=(20, 8))
fig.suptitle('Transformation Analysis',fontsize=20)
sns.distplot(train['LoanAmount'],ax=axes[0,0],color='red')
sns.boxplot(train['LoanAmount'],ax=axes[1,0],color='orange')
sns.distplot(norm_func(train['LoanAmount']),ax=axes[0,1],color='green')
sns.boxplot(norm_func(train['LoanAmount']),ax=axes[1,1],color='orange')
sns.distplot(norm_func(np.power(train['LoanAmount'],1/3)),ax=axes[0,2],color='green')
sns.boxplot(norm_func(np.power(train['LoanAmount'],1/3)),ax=axes[1,2],color='orange')
sns.distplot(norm_func(np.log(train['LoanAmount'])),ax=axes[0,3],color='green')
sns.boxplot(norm_func(np.log(train['LoanAmount'])),ax=axes[1,3],color='orange')
plt.savefig('TransformationAnalysis_LoanAmount.png')
```

Treating Outliers



Feature Engineering

Total Income

- Applicant Income + Coapplicant Income

EMI = LoanAmount / Loan_Amount_Term

- But EMI also has interest rate and we consider 10% for monthly (10/100)/12 = 0.0083
  - EMI = $\frac{P \times r \times (1+r)^n}{(1+r)^{n-1}}$
    - P = Principal Loan Amount
    - r = Rate of Interest
    - n = Tenure of repayment or Loan_Amount_Term

New Possible Features Generated

Credit_History wise Total Income

- Group Credit_History with sum of Total Income

Dependents wise LoanAmount

- Group Dependents with sum of LoanAmount

Debt to Income Ratio

- Ratio of LoanAmount to TotalIncome

EMI to Loan_Amount_Term

- Ratio of EMI to Loan_Amount_Term

Property Grouping for Income Ratio

- Group Property with Mean of Income Ratio

Source for Interest Rate : https://www.hdfc.com/housing-loans/home-loan-interest-rates

Source for EMI Formula : https://www.bajajfinserv.in/home-loan-emi-calculator

Category to Numerical Encoding

Label Encoding

Target Variable

- Loan Status

Label Encoding Target Feature

```
le = LabelEncoder()
train['Loan_Status'] = le.fit_transform(train['Loan_Status'])
```

executed in 15ms, finished 13:09:39 2021-05-20

One-Hot Encoding

Non-Ordinal and Independent Variables

- Gender
- Married
- Education
- Self-employed

One-Hot Encoding Independent Non-Ordinal Features

```
train_ohe = train.loc[:,['Gender','Married','Education','Self_Employed']]
train_ohe = pd.get_dummies(train_ohe)
test_ohe = test.loc[:,['Gender','Married','Education','Self_Employed']]
test_ohe = pd.get_dummies(test_ohe)

train= pd.concat([train_ohe,train],axis=1)
test= pd.concat([test_ohe,test],axis=1)
```

executed in 47ms, finished 13:09:43 2021-05-20

Feature Engineering

- After Finding Features Train set has 25 Columns and Test has 24 respectively.
- But After finding the ranks of Best Features we select Top 15 Features for building Classification Model.

Unbalanced Classes

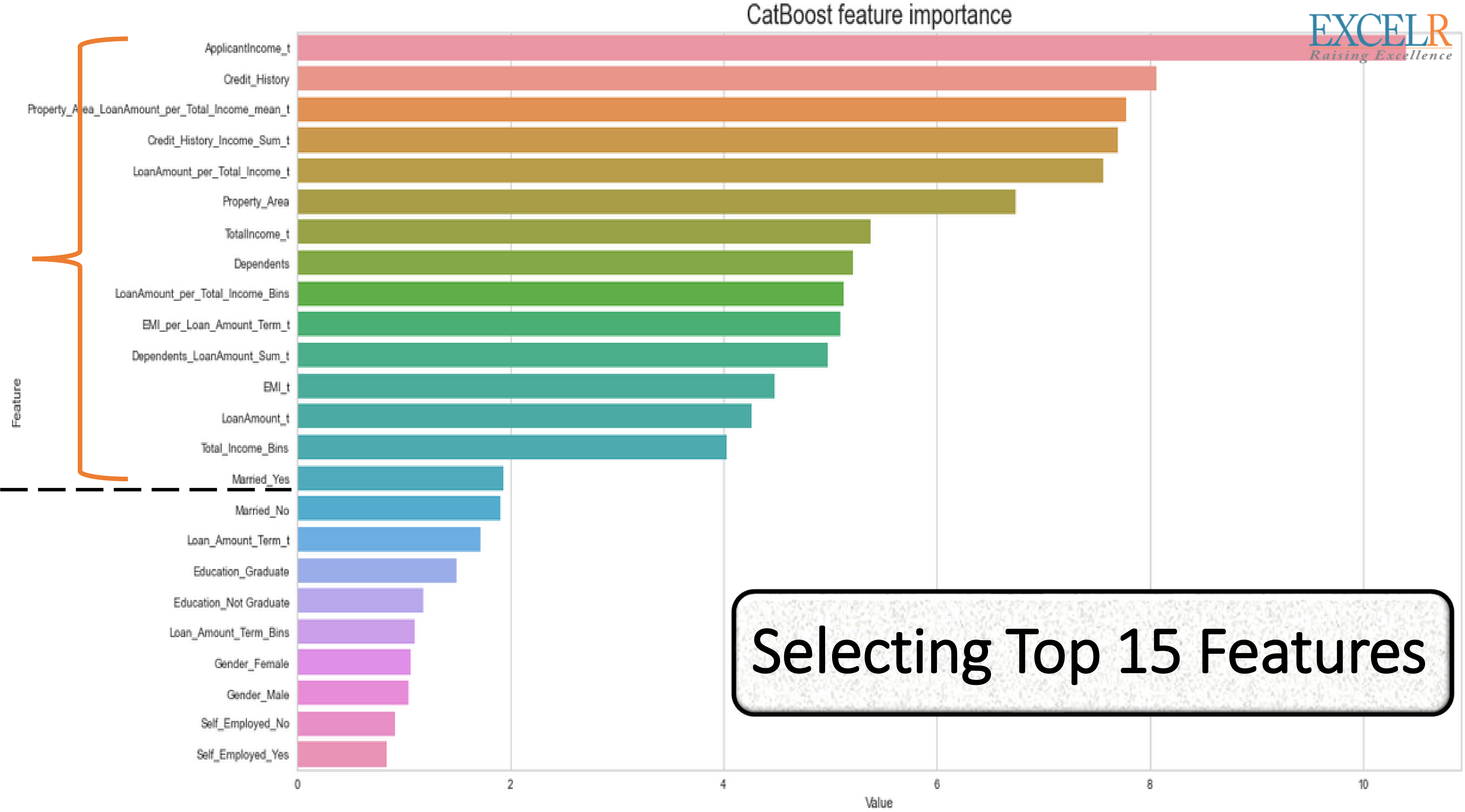
We use SMOTE analysis and do Oversampling to give model equal chances to classify data.

Modeling with Top 15 Features & Up-sampling using SMOTE for solving Unbalanced Data

```
X1 = train1.iloc[:,:-1]
y1 = train1.iloc[:,-1]
```

```
smk = SMOTETomek(random_state=101)
X1_res,y1_res = smk.fit_resample(X1,y1)
X1.shape,y1.shape,X1_res.shape,y1_res.shape
```

```
((614, 15), (614,), (752, 15), (752,))
```



Selecting Top 15 Features

Model Summary

Sr. No.	Classification Models	Accuracy			ROC-AUC Score	Classification Report							
		Stratified K-Fold	Train	Validation		Accuracy	Precision		Recall		F-1 Score	
							N	Y	N	Y	N	Y
1	Logistic Regression	0.77	0.78	0.74	0.77	0.74	0.69	0.76	0.53	0.86	0.60	0.81
2	Decision Tree	0.78	0.79	0.76	0.72	0.76	0.70	0.78	0.58	0.86	0.63	0.82
3	CatBoost Classifier	0.76	0.99	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
Modeled with Top 15 Features (Feature Engineering)												
4	Logistic Regression	0.73	0.73	0.78	0.85	0.77	0.80	0.76	0.69	0.85	0.74	0.80
5	CatBoost Classifier	0.85	1.00	0.84	0.93	0.84	0.77	0.93	0.93	0.77	0.84	0.84
6	Decision Tree	0.78	0.81	0.76	0.75	0.76	0.74	0.78	0.76	0.77	0.75	0.77
7	Bagging Classifier	0.79	0.84	0.80	0.85	0.80	0.79	0.81	0.79	0.81	0.79	0.81
8	Random Forest Classifier	0.79	0.84	0.80	0.87	0.80	0.82	0.79	0.73	0.86	0.77	0.82
9	AdaBoost Classifier	0.78	0.84	0.76	0.80	0.76	0.73	0.79	0.77	0.75	0.75	0.77
10	SVM	0.83	0.96	0.77	0.87	0.77	0.70	0.87	0.89	0.68	0.78	0.76
11	Stacking	0.82	0.85	0.78		0.78	0.76	0.80	0.77	0.79	0.77	0.80
12	Extra Tree Classifier	0.80	1.00	0.87	0.93	0.87	0.84	0.91	0.90	0.85	0.87	0.88
13	Gradient Boost Classifier	0.85	1.00	0.79	0.92	0.79	0.72	0.89	0.90	0.70	0.80	0.79
14	Naive Bayes Classifier	0.72	0.72	0.77	0.84	0.78	0.89	0.73	0.60	0.94	0.72	0.82
15	KNN Classifier	0.77	0.81	0.79	0.85	0.79	0.76	0.82	0.80	0.78	0.78	0.80
16	XG Boost Classifier	0.77	0.80	0.82	0.87	0.82	0.81	0.83	0.80	0.84	0.81	0.83
17	Neural Network	-	0.91	0.79	-	-	-	-	-	-	-	-

Challenges Faced

- Finding New Features
- Data Processing
- Deviation in Accuracy
- Hardware Limitations

Bank Websites, Bank Employee

Studying Bank Terms

Overcoming Challenge

Hyperparameter Tuning

Checking Accuracy on Analytics Vidhya Portal

Loan Prediction

Online 26-05-2016 12:01 AM to 30-06-2021 11:59 PM

ENDS IN

40 DAYS 5 HOURS 34 MIN

68742
Registered

Practice Problem
Prizes

Registered

#		Name	Score	Submission Trend	Participant's approach	AV Rank
363	A	amit0902	0.7986111111		Add approach	11160

Next Phase

EXCELR
Raising Excellence

Work on Comments given by Mentor

Try Improving Model Accuracy

Model Deployment

Model Deployment Screen