LOAN APPLICATION BLOG

INTRODUCTION

What is a Loan?

A loan is a sum of money that one or more individuals or companies borrow from banks or other financial institutions so as to financially manage planned or unplanned events. In doing so, the borrower incurs a debt, which he must pay back with interest and within a given period.

Who Applies for Loan?

Throughout the world people apply loans for various reasons, they can be personal loan, home loan, student loan, etc. How do they decide they are eligible? And based on what criteria are they eligible?

There are many criterions as to which the company or firm decides to approve the applicant or not, a few of the criterions are present in our dataset.

We have a dataset for which the applications were either approved or not, according to the criterion of the applicant given below:

Loan_ID: ID of the Applicant
Gender: Male / Female
Married: Yes / No
Dependents: Number of dependents
Education: Graduated / Ungraduated
Self_Employed: Yes / No
ApplicantIncome: Applicant Income
CoapplicantIncome: Co-Applicant Income
LoanAmount: Amount on Loan
Loan_Amount_Term: Time to pay back the loan in Months
Credit_History: Credit History meet guidelines, Yes / No
Property_Area: Urban/ Semi Urban/ Rural
Loan_Status: Approved or Not (Yes / No)

Exploratory Data Analysis

I have used seaborn for and Visualization and pandas for manipulation of Data.
Let's import the necessary libraries for plotting graphs, manipulating of data and importing the data set as given below:

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
import pandas as pd
df=pd.read_csv('Loan Application Status Prediction.csv')
```

```
df
```

Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area
Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban
Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural
Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban
Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban
Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban
...	...	...	...	...	...	...	...	...	...	...
Female	No	0	Graduate	No	2900	0.0	71.0	360.0	1.0	Rural
Male	Yes	3+	Graduate	No	4106	0.0	40.0	180.0	1.0	Rural
Male	Yes	1	Graduate	No	8072	240.0	253.0	360.0	1.0	Urban
Male	Yes	2	Graduate	No	7583	0.0	187.0	360.0	1.0	Urban
Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0	0.0	Semiurban

614 rows × 13 columns

Data-Set Information

```
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 
```

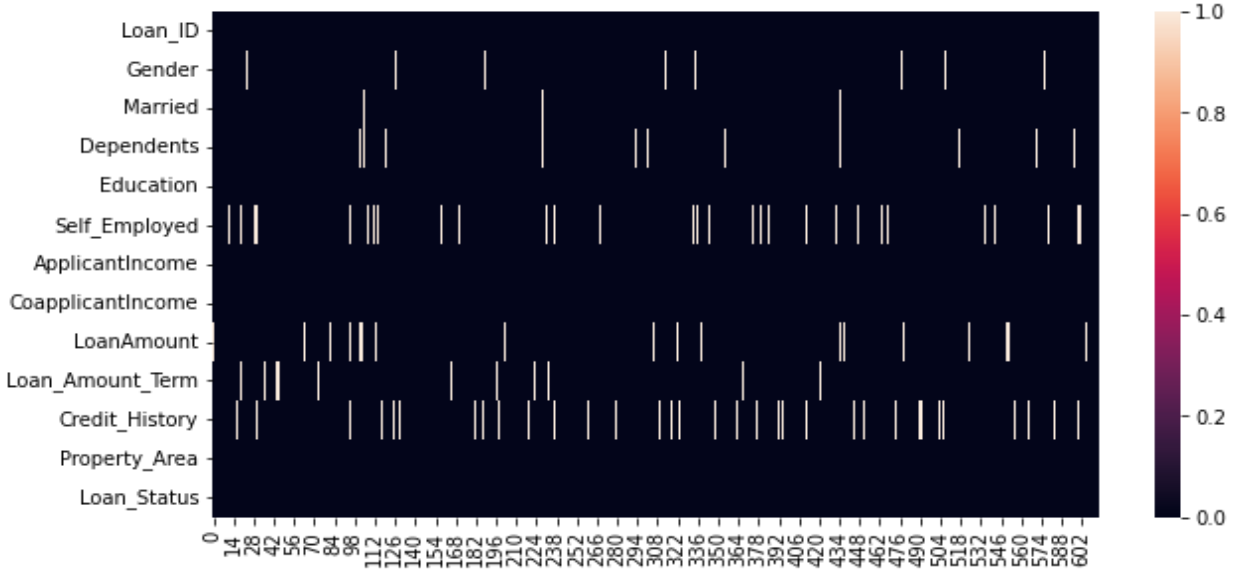
There are a total of 614 Rows and 13 Columns of which 8 are ‘Object’, 4 are ‘float’ and 1 is an ‘int’ datatype. We can notice there are missing values in these columns.

Unique Values For Object Columns

```
Gender : ['Male' 'Female' nan]
Married : ['No' 'Yes' nan]
Dependents : ['0' '1' '2' '3+' nan]
Education : ['Graduate' 'Not Graduate']
Self_Employed : ['No' 'Yes' nan]
Property_Area : ['Urban' 'Rural' 'Semiurban']
Loan_Status : ['Y' 'N']
```

Null Values

First let's check in which columns there are Null Values present in the dataset:

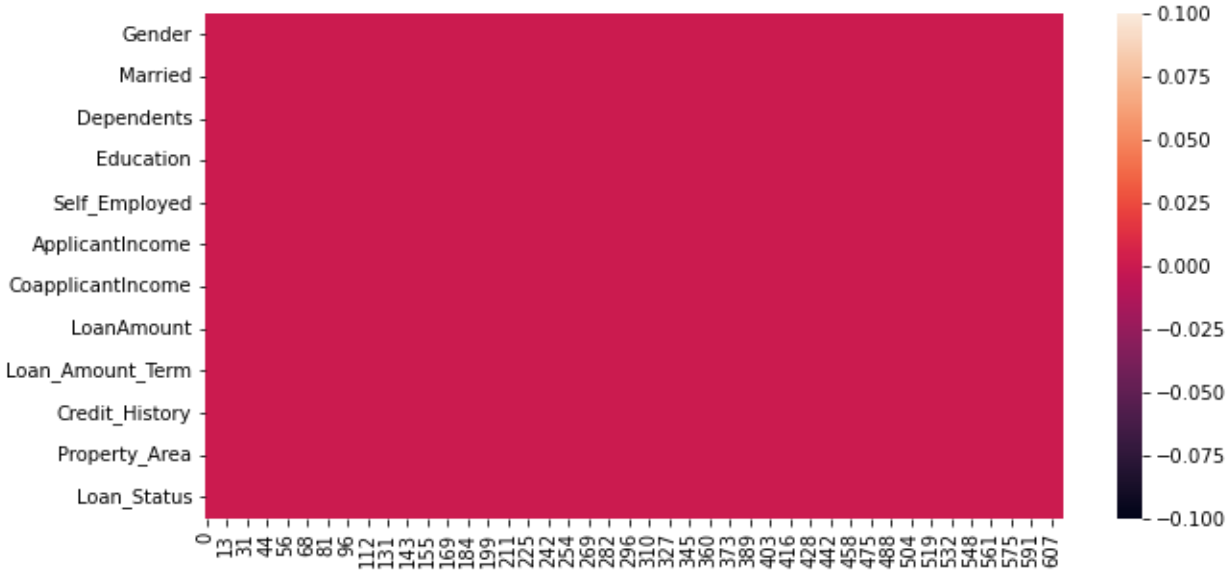



```
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           22
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status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

From the plot diagram and information, we can see few missing values in columns: Gender, Married, Dependents, Self_Employed, LoanAmount, Loan_Amount_Term, Credit_History. There are different ways of solving missing values, we can either drop them, fill according to most occurred value or mean etc.

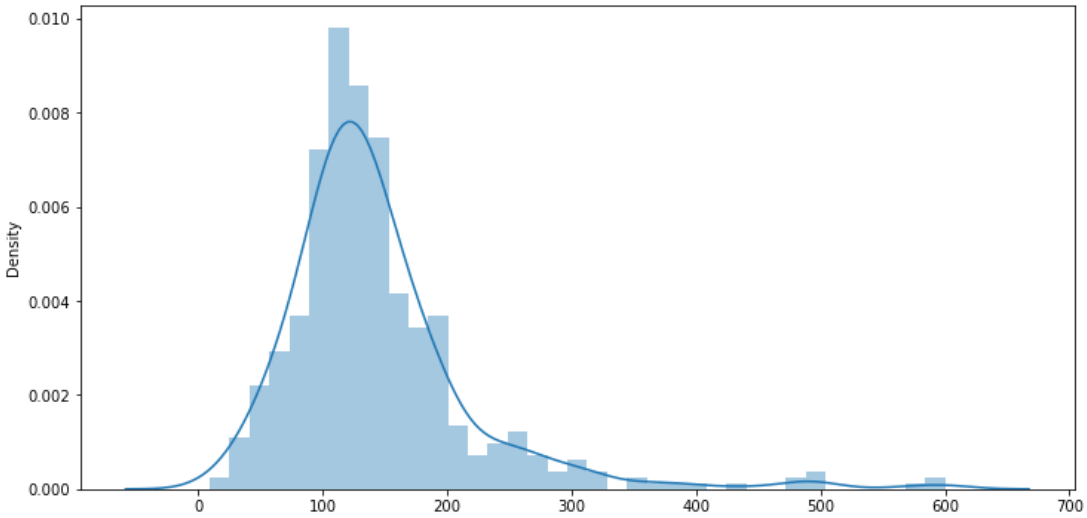
I have replaced the missing values of LoanAmount with mean of it and Loan_Amount_Term with its mode and dropped the rest of the missing values as well as dropped 'Loan_ID' column since it does not have a relation with eligibility of applicant as shown below.

```
df['LoanAmount']=df['LoanAmount'].replace(np.nan,np.mean(df['LoanAmount']))
df['Loan_Amount_Term']=df['Loan_Amount_Term'].replace(np.nan,stat.mode(df['L
oan_Amount_Term']))
df.drop('Loan_ID',axis=1,inplace=True)
df=df.dropna()
```



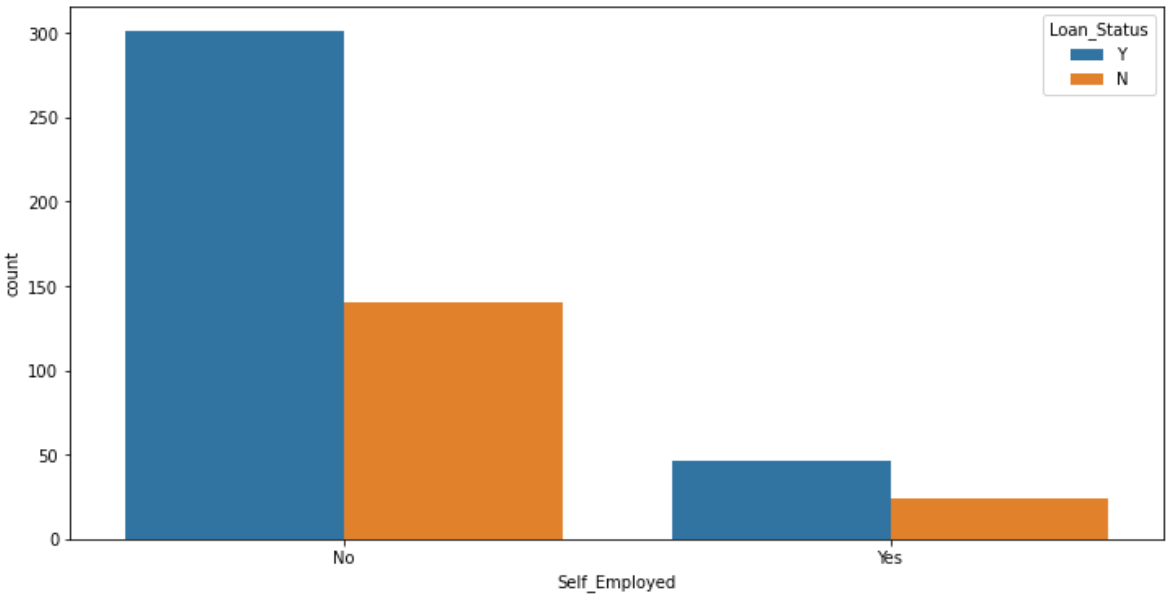
Graphical Visualization of Data

Loan Amount

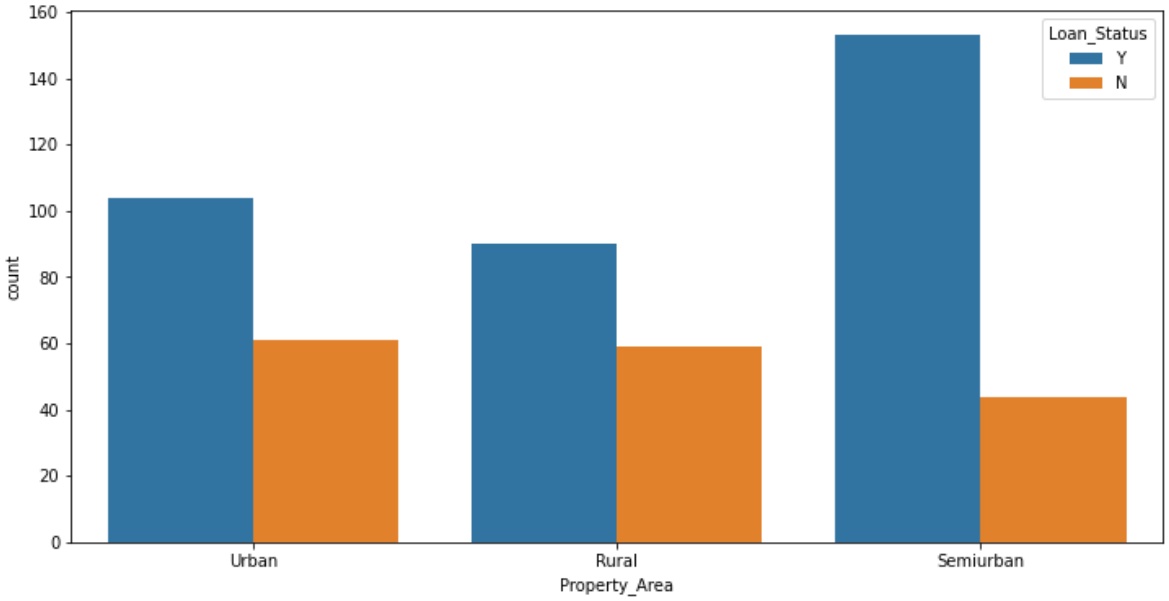


This is a graphical representation of **"Loan Amount"** taken by the Applicants, we can see that it is Right Skewed plot of range 100~200. We can also see that there are very few that apply for a loan of 500 and 600, In this graph there are outliers noticeable between 400~600.

Self Employed



Property Area



Converting Datatypes

Now we will convert all the categorical variables into numbers using LabelEncoder in sklearn.

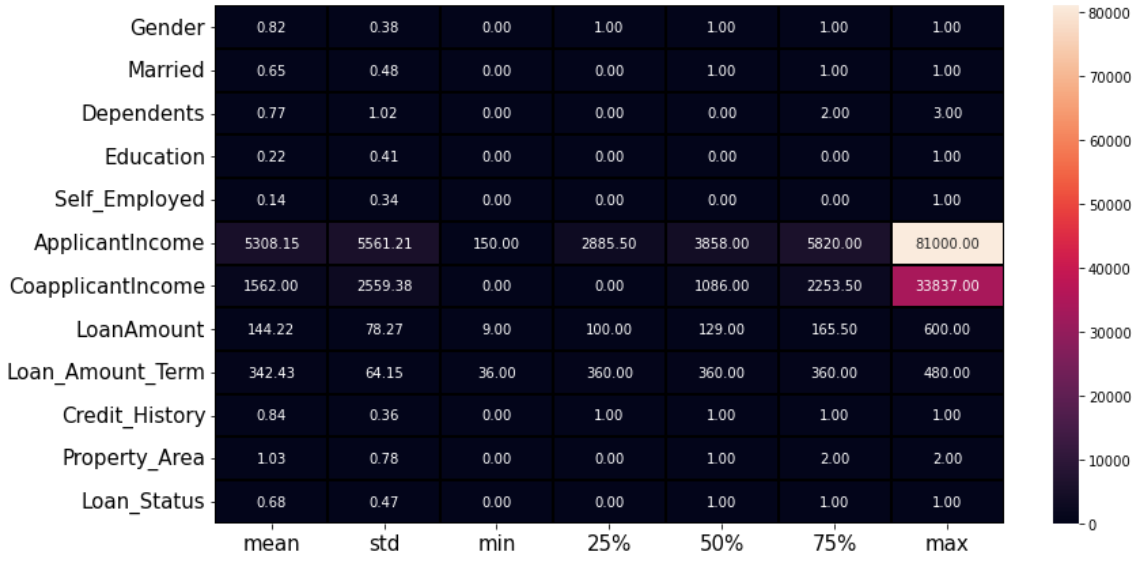
```
from sklearn.preprocessing import LabelEncoder
enc=LabelEncoder()
col=['Gender', 'Married', 'Dependents', 'Education','Self_Employed',
'Property_Area', 'Loan_Status']
for i in col:
    df[i]=enc.fit_transform(df[i].values.reshape(-1,1))
```

```
# Converting 'float' to 'int'
df['Credit_History'] = df['Credit_History'].astype(int)
```

Here we have converted columns ('Gender', 'Married', 'Dependents', 'Education','Self_Employed', 'Property_Area', 'Loan_Status') to numbers for model training and further data analysis.

Data Description:

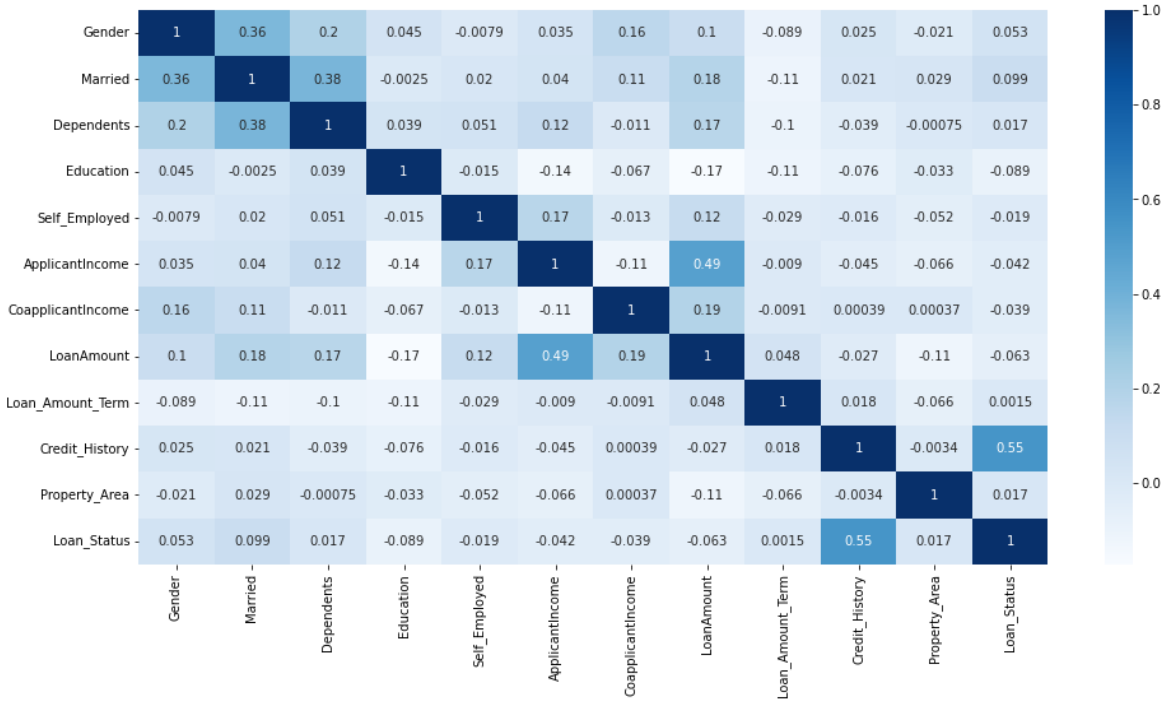
	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
count	511.000000	511.00000	511.000000	511.000000	511.000000	511.000000	511.000000	511.000000	511.000000	511.000000	511.000000	511.000000
mean	0.821918	0.64775	0.769080	0.215264	0.136986	5308.146771	1561.997886	144.220804	342.434442	0.843444	1.031311	0.679061
std	0.382957	0.47814	1.020142	0.411408	0.344170	5561.209487	2559.383166	78.266321	64.150735	0.363738	0.784031	0.467295
min	0.000000	0.00000	0.000000	0.000000	0.000000	150.000000	0.000000	9.000000	36.000000	0.000000	0.000000	0.000000
25%	1.000000	0.00000	0.000000	0.000000	0.000000	2885.500000	0.000000	100.000000	360.000000	1.000000	0.000000	0.000000
50%	1.000000	1.00000	0.000000	0.000000	0.000000	3858.000000	1086.000000	129.000000	360.000000	1.000000	1.000000	1.000000
75%	1.000000	1.00000	2.000000	0.000000	0.000000	5820.000000	2253.500000	165.500000	360.000000	1.000000	2.000000	1.000000
max	1.000000	1.00000	3.000000	1.000000	1.000000	81000.000000	33837.000000	600.000000	480.000000	1.000000	2.000000	1.000000



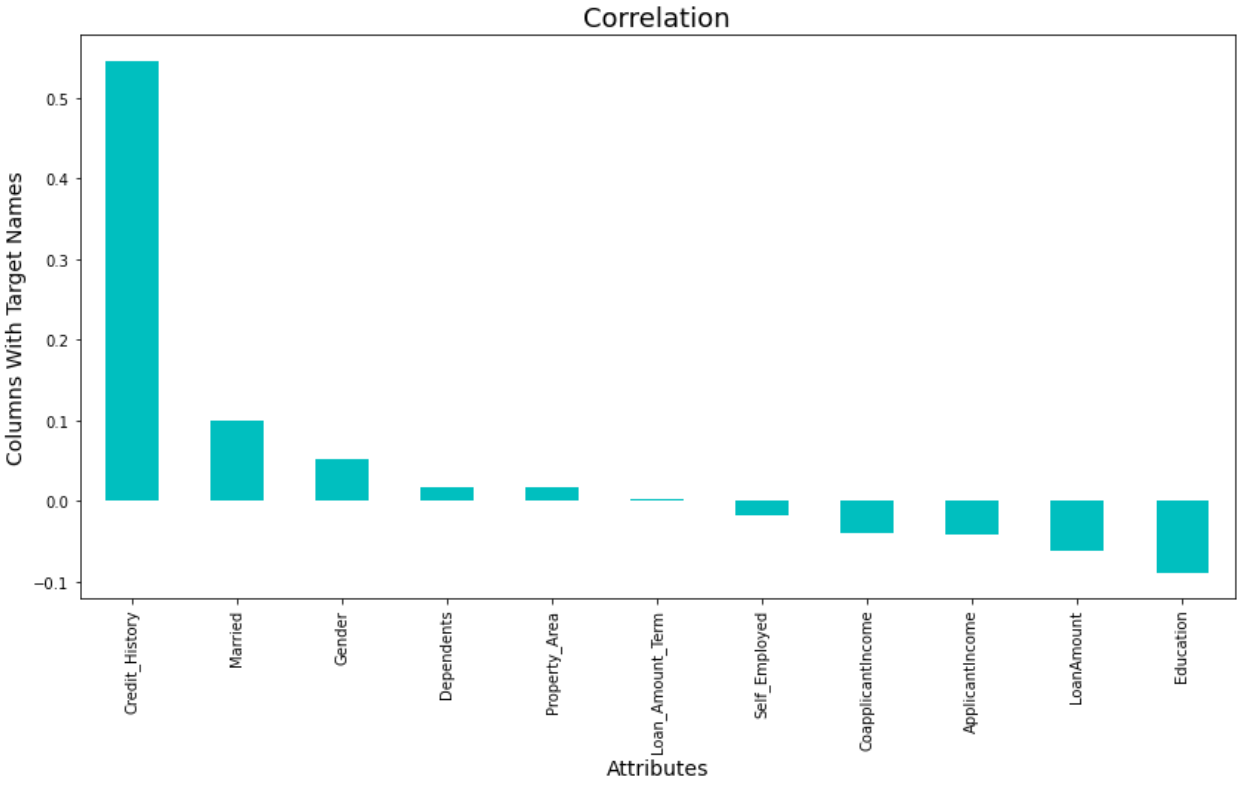
From the plot and Data Frame we can study and analyze the statistical structure of the dataset, we can see outliers present in Applicant Income since mean and Q3(3rd Quartile) are close to each other and difference between Q3 and mx is very large, and outliers is also present in Co-applicant Income where max far more than Q3 compared to Q2 and Q3, and in LoanAmount also is the same case as the previous ones.

Credit_History is more than 0.5 i.e 0.84 it means there is more occurrence 1 (Credit History meet guidelines: Yes), since there are only to values 0,1.
There are married applicants of about 65% who apply for loans compared to unmarried applicants.
But the are only 22% of the applicants that have Graduated.

Correlation of Data:



In this Correlation plot diagram we can study the relationship between columns, the columns Loan_Status and Credit_History have the highest correlation to each other of 0.55, and correlation of LoanAmount and ApplicantIncome being 0.49.



The above plot is the correlation between Loan_Status (Target Column) and the other columns (Feature columns) with Credit_History having the high positive correlation and Education having a negative correlation with Loan_Status column.

Removing Outliers

As we have seen in the previous Graphical diagram and Data Frames, there are outliers present that can bring errors in Model Training. We can use import 'Z-score' to remove outliers as given below:

```
from scipy.stats import zscore
z=np.abs(zscore(df.iloc[:,:11]))
threshold=3
df_new=df[(z<3).all(axis=1)]
x=df_new.drop(['Loan_Status'],axis=1)
y=df_new['Loan_Status']
```

As seen in the code, after removing outliers we have 480 rows left, later we separated the dataset into Feature columns as 'X' and Target column as 'Y'. Next is balancing the data.

Data Balancing

```
import sklearn
from imblearn.over_sampling import SMOTE
smt=SMOTE()
x_new,y=smt.fit_resample(x,y)
x=pd.DataFrame(x_new,columns=x.columns)
```

Since there was a difference of more than 50% between Accepted Applicants and Rejected Applicants, I have performed Data Balancing operation.

Min-Max Scaling

```
from sklearn.preprocessing import MinMaxScaler
scale=MinMaxScaler()
x_new=scale.fit_transform(x)
x=pd.DataFrame(x_new,columns=x.columns)
```

By importing and using MinMaxScaler library I have decreased the range of data by transforming the feature columns, due to few columns have a large range.

Skew

```
Gender              -1.214500
Married             -0.182990
```

```
Dependents           1.204683
Education            1.627463
Self_Employed        2.590842
ApplicantIncome      2.093491
CoapplicantIncome    1.230633
LoanAmount           1.065361
Loan_Amount_Term    -1.694785
Credit_History      -0.828117
Property_Area        0.057725
```

There are columns with positive skewness and negative skewness of which Self_Employed and ApplicantIncome have the highest positive skewness. Since there is skewness we will import power_transform to change the skewness in the feature columns.

```
from sklearn.preprocessing import power_transform
x_new=power_transform(x)
x=pd.DataFrame(x_new,columns=x.columns)
```

SELECTING BEST MODEL

We will apply different models for training and testing to compare with using Accuracy Score, Classification report, Confusion Matrix and Cross Validation to get the best possible working model.
First let us import the necessary libraries:

```
from sklearn import metrics
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report,confusion_matrix
from sklearn.model_selection import train_test_split
```

Now lets try different Classifier Model.

Decision Tree Classifier

```
Best accuracy:  87.27272727272727  on Random State:  11
[[68 20]
 [18 59]]
              precision    recall  f1-score   support

           0       0.79      0.77      0.78        88
           1       0.75      0.77      0.76        77

    accuracy                           0.77       165
   macro avg       0.77      0.77      0.77       165
weighted avg       0.77      0.77      0.77       165
Cross validation score is:  0.78
```

Random Forest Classifier

```
Best accuracy:  90.30303030303031  on Random State:  67
```

```
[[68 20]
 [11 66]]
              precision    recall  f1-score   support

           0       0.86      0.77      0.81        88
           1       0.77      0.86      0.81        77

    accuracy                           0.81       165
   macro avg       0.81      0.81      0.81       165
weighted avg       0.82      0.81      0.81       165
Cross validation score is:  0.83
```

K-Neighbor Classifier

```
Best accuracy:  84.24242424242424  on Random State:  79
[[62 26]
 [13 64]]
              precision    recall  f1-score   support

           0       0.83      0.70      0.76        88
           1       0.71      0.83      0.77        77

    accuracy                           0.76       165
   macro avg       0.77      0.77      0.76       165
weighted avg       0.77      0.76      0.76       165
```

Support Vector Classifier

```
Best accuracy:  85.45454545454545  on Random State:  79
[[60 28]
 [ 7 70]]
              precision    recall  f1-score   support

           0       0.90      0.68      0.77        88
           1       0.71      0.91      0.80        77

    accuracy                           0.79       165
   macro avg       0.80      0.80      0.79       165
weighted avg       0.81      0.79      0.79       165
Cross validation score is:  0.79
```

Logistic Regression

```
Best accuracy:  86.66666666666667  on Random State:  83
[[63 25]
 [ 8 69]]
              precision    recall  f1-score   support

           0       0.89      0.72      0.79        88
           1       0.73      0.90      0.81        77
```

```
    accuracy                           0.80       165
   macro avg       0.81      0.81      0.80       165
weighted avg       0.82      0.80      0.80       165
Cross validation score is:  0.80
```

Gaussian Naive Bayes

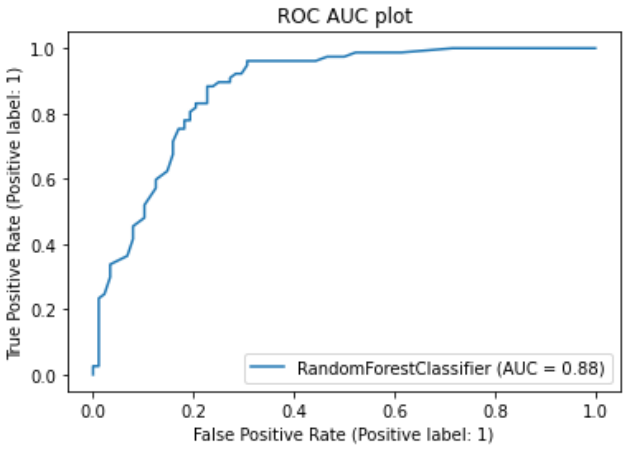
```
Best accuracy:  86.66666666666667  on Random State:  83
[[60 28]
 [ 5 72]]
              precision    recall  f1-score   support

           0       0.92      0.68      0.78        88
           1       0.72      0.94      0.81        77

    accuracy                           0.80       165
   macro avg       0.82      0.81      0.80       165
weighted avg       0.83      0.80      0.80       165
Cross validation score is:  0.79
```

CONCLUSION

After analyzing the data from different views using plot diagram, Statistical data and Correlation, later transforming the data for changing datatypes from categorical to numbers, removing outliers, balancing the data using Scaler and adjusting skewness. The next stage that was model selection, we applied different classifiers to select the best one out of it.
And so as Random Forest Classifier is seen as the best model working with Accuracy Score: 90%, Cross Validation Score: 83% and AUC Curve: 88% as shown below.



Since Random Forest Classifier model is concluded as best model, we will save the model in the below code importing 'joblib'.

```
import joblib
joblib.dump(rf,'Census Income Model.pkl')
```