DEPARTMENT OF STATISTICS, UNIVERSITY OF KALYANI

PREDICTION FOR LOAN APPROVAL USING MACHINE LEARNING ALGORITHM

SUBMITTED AS DISSERTATION PAPER FOR THE FINAL SEMESTER EXAMINATION OF THE 2-YEAR M.Sc. IN STATISTICS COURSE

BY

Tapomayee Basu

REG NO.: 100417 of 2020 - 2021

ROLL NO.: 96/STA No. 200033

UNDER THE GUIDANCE OF

Mr. Taranga Mukherjee

Faculty

Department of Applied Statistics

Maulana Abul Kalam Azad University of Technology (WBUT)

ACKNOWLEDGEMENT

First of all, I would like to express my sincere gratitude to my guide Prof. Taranga Mukherjee for helping me since the very beginning, starting right from – enriching me with ideas of interest and introducing me to any and every necessary supply that would require attention while making of this project. Yet another motivation of acknowledgement on his name would be the fact that Sir has always been a philosophical guide most of all. We are thankful to you sir.

I would like to thank our Head of the Department Prof. Chandranath Paul for his consistent support and guidance during the running of our course. Thank you, sir, for bearing with us and helping us out with doubts in and out of the time table.

Furthermore, I would like to thank the rest of our department professors. Prof. Natasha Dasgupta ma'am for introducing us to projects and guiding us through our presentation, Ganesh Dutta sir for giving us projects to be solved.

To conclude, I cannot forget to thank my family and friends for all the unconditional support in this very intense academic year. I certainly did not have enough time, but with all the collective efforts from you all, I have been able to make through it.

CONTENTS

ABSTRACT

In the banking industry the main source of income lies on its credit line. That is a bank earns interests off of the loans which it credits. A bank's profit and loss depend on a very large extent on loans. So, it becomes very important for this particular industry to study its customers for steady analysis of who is eligible to obtain the loan and who is not.

Loan Acceptance is a crucial step here. The system either approves or rejects the loan applications. It is incredibly difficult to predict if a customer will repay the loan or not. So, a company wants to automate the loan eligibility procedure (in real time) by using the basic information of a client submitted into an online application form.

The objective of this project is to achieve two goals. They are as follows:

1. Identification of significant characteristics that indicate a borrower's ability to repay the loan.

2. Identifying the optimal model(s) for assessing credit risk and approving the loan application.

The major goal of this study is to determine whether or not designating a loan to a certain person is safe. In modern era of Data Science and Machine Learning, it is easy for us to decide the aforementioned problem. This project gives us a complete view of how to solve the problem using very sophisticated Machine Learning Algorithm. Here we want to classify the loan applicants based on whether loan has been approved or not. We perform exploratory data analysis and build models using various Classification Algorithm to get a model having maximum accuracy.

INTRODUCTION

In finance, a loan is the lending of money by one or more individuals, organizations, or other entities to other individuals, organizations etc. The recipient (i.e., the borrower) incurs a debt and is usually liable to pay interest on that debt until it is repaid as well as to repay the principal amount borrowed.

A loan is a bank's main source of revenue. Even though the bank accepts the loan following a lengthy verification and testimony process, there is no guarantee that the chosen candidate is the right one. When done manually, this operation takes a long time. We can predict whether a given hopeful is safe or not, and the entire testimonial process is automated using machine literacy. Loan Prognostic is beneficial to both bank retainers and hopefuls.

Considering the two most important banking issues amongst others:

1) What is the borrower's risk level?

2) Given the danger, should we lend to the borrower?

The interest rate, together with other factors (such as the time value of money), assesses the borrower's riskiness, the higher the interest rate, the riskier the borrower. Based on the interest rate, we will determine whether the applicant is eligible for the loan.

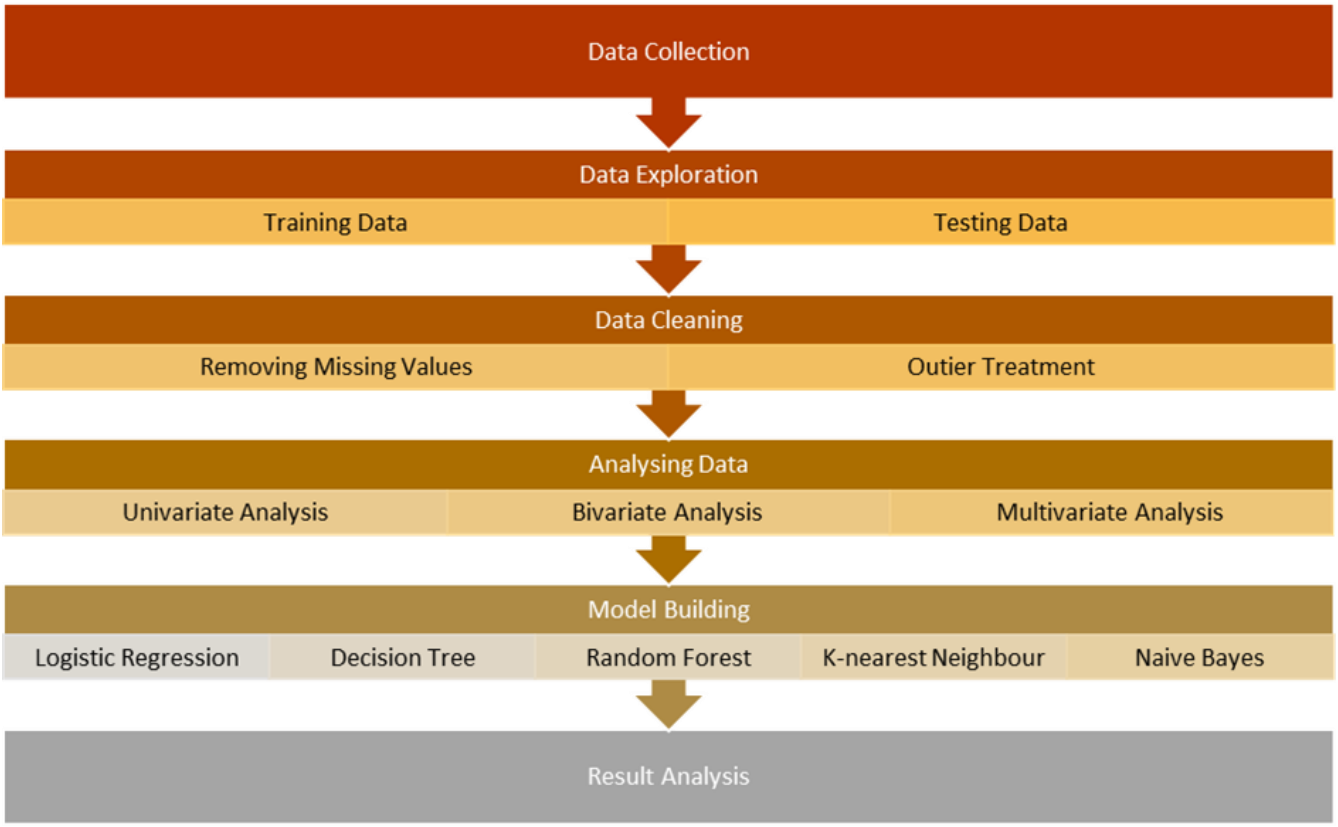
Lenders provide loans to creditors in exchange for interest-bearing repayment guarantees. The lender only gets paid (interest) if the borrower pays back the loan.

Customers are given loans by banks in exchange for a guarantee of payback. In case of inability of repayment of loans, the bank keeps insurance to reduce the risk of collapse. When there were a significant number of loan applications, manual techniques were usually effective, but they were insufficient. Making a decision at the time would take a long time. As a result, the Machine Learning Model for Loan Prediction can be used to assess a customer's loan condition and develop plans and to make the job of a bank management easier. This model extracts and introduces the key characteristics of a borrower that determine the Loan Status of the consumer.

Several Machine Learning techniques used for Loan Approval Prediction are:

1. Logistic Regression
2. Decision Tree
3. Random Forest
4. K Nearest Neighbor
5. Naïve Bayes Classifier

METHODOLOGY



MACHINE LEARNING AND CONCEPTS

Classification is the process of categorizing a given set of data into classes. In this case we have binary classification. In Machine Learning, we frame the problem, collect and clean the data, add some necessary feature variables (if any), train the model, measure its performance, improve it by using some cost function, and then it is ready to deploy. Here we have used Logistic Regression, Decision Tree, Random Forest, K Nearest Neighbor, Naïve Bayes Classifier models for predictive analysis which will be discussed in details later.

Evaluation metrics for classification

The process of model building is not complete without the evaluation of model performance. Suppose we have the predictions from the model, how can we decide whether the predictions are accurate? We can plot the results and compare them with the actual values i.e., calculate the distance between the predictions and actual values. The lesser this distance more accurate will be the predictions. Since this is a classification problem, we can evaluate our models using any one of the following evaluation metrics:

The **Confusion Matrix** is a way to evaluate the performance of a classifier. In predictive analytics, it is a two-by-two table that tells us the rate of false positives, false negatives, true positives and true negatives for a test or predictor. We can make a confusion matrix if we know both the predicted values and the true values for a sample set. Since it shows the errors in the model performance in the form of a matrix, hence also known as an E**rror** M**atrix.**

		Predicted Class: Positive	**Predicted Class: Negative**	
Actual Class	**Positive**	True Positive (TP)	False Negative (FN) **Type II Error**	**Sensitivity** $\frac{TP}{(TP+FN)}$
	Negative	False Positive (FP) **Type I Error**	True Negative (TN)	**Specificity** $\frac{TN}{(TN+FP)}$
		Precision $\frac{TP}{(TP+FP)}$	**Negative Predictive Value** $\frac{TN}{(TN+FN)}$	**Accuracy** $\frac{TP+TN}{(TP+TN+FP+FN)}$

Where,

- **True Negative:** Model has given prediction No, and the real or actual value was also No.
- **True Positive:** The model has predicted Yes, and the actual value was also True.
- **False Negative:** The model has predicted No, but the actual value was Yes. It is also called as **Type-II error**.
- **False Positive:** The model has predicted Yes, but the actual value was No. It is also called a **Type-I error.**

A **Classification report** is used to measure the quality of predictions from a classification algorithm. How many predictions are True and how many are False? More specifically, True Positives, False Positives, True negatives and False Negatives are used to predict the metrics of a classification report as shown below. The report shows the main classification metrics precision, recall and f1-score on a per-class basis. The metrics are calculated by using true and false positives, true and false negatives.

Accuracy: When we talk of accuracy, we are referring to how close the measured value (what we are predicting) is to the known values. To calculate the accuracy of a model from our confusion matrix we would sum the correct answers (TP + TN) and divide it by the total number of instances (TP + TN + FP + FN).

Precision: Precision, also known as positive predictive value, informs us of the amount of actual positive labels from all of the labels our classifier has labelled as positive.

Recall: Recall, also known as sensitivity or the true positive rate (TPR), informs us of the number of positive labels that our classifier correctly labelled as positive.

F1 Score: It's quite rare that precision and recall are discussed in isolation, and they often tend to have an inverse relationship, where optimizing for one metric would reduce the other. In situations where we need to strike a balance between precision and recall, a better-known metric to look to is the F1-score, also referred to as the F-measure. The F1 score is a weighted harmonic mean of precision and recall such that the best score is 1.0 and the worst is 0.0. Generally speaking, F1 scores are lower than accuracy measures as they embed precision and recall into their computation.

F1 Score = 2*(Recall * Precision) / (Recall + Precision)

EXPLORATORY DATA ANALYSIS

DATA COLLECTION:

We have used a public dataset for loan prediction from the website Kaggle. The link is given below.

(https://www.kaggle.com/datasets/altruistdelhite04/loan-prediction-problem-dataset)

The dataset utilized consists of 13 features divided into two sets: a training set with 642 rows and a testing set with 276 rows. The features of the dataset are depicted in the table below, along with their proper descriptions:

VARIABLE	DATA TYPE	DESCRIPTION
Loan_ID	Catogerical(non-numeric)	Unique loan ID
Gender	Catogerical(non-numeric)	Male/Female
Married	Catogerical(numeric)	Yes/No (marital status)
Dependants	Catogerical(numeric)	No. Of dependants
Education	Catogerical(non-numeric)	Graduate/Not graduate
Self_Employed	Catogerical(non-numeric)	Yes/No
Applicant_Income	Numeric feature	Applicant income
Coapplicant-Income	Numeric feature	Co-applicant income
Loan_Amount	Numeric feature	Loan amount in thousands
Loan_Amount_Term	Numeric feature	Term of loan in months
Credit_History	Catogerical(numeric)	0/1 (previous credit history)
Property_Area	Catogerical(non-numeric)	Urban/semi-Urban/rural
Loan_Status	Catogerical(non-numeric)	Yes/No (Our target variable)

First 5 rows of the dataset-

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y

DATA EXPLORATION:

- **Importing Libraries:** We start with importing the necessary libraries and packages as follows:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
import missingno as msno
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report, roc_auc_score
from sklearn.model_selection import cross_val_score

import warnings
warnings.filterwarnings('ignore')
```

- **Loading dataset:** Then we load the dataset. Our dataset has 981 rows and 13 columns just as we have already mentioned.

```
df = pd.read_csv("C:\\Users\\Downloads\\Loan Predicton Data.csv")
df
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	Applicant_Income	Coapplicant_Income	Loan_Amount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	LP001002	Male	No	0.0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	LP001003	Male	Yes	1.0	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
2	LP001005	Male	Yes	0.0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
3	LP001006	Male	Yes	0.0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
4	LP001008	Male	No	0.0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y
...	...	...	...	...	...	...	...	...	...	...	...	...	...
976	LP002971	Male	Yes	3.0	Not Graduate	Yes	4009	1777.0	113.0	360.0	1.0	Urban	Y
977	LP002975	Male	Yes	0.0	Graduate	No	4158	709.0	115.0	360.0	1.0	Urban	Y
978	LP002980	Male	No	0.0	Graduate	No	3250	1993.0	126.0	360.0	NaN	Semiurban	Y
979	LP002986	Male	Yes	0.0	Graduate	No	5000	2393.0	158.0	360.0	1.0	Rural	Y
980	LP002989	Male	No	0.0	Graduate	Yes	9200	0.0	98.0	180.0	1.0	Rural	Y

981 rows × 13 columns

From what we observe from the data, we see that we have 13 features out of which we have 12 independent variables and 1 dependent variable. ‘Loan_Status’ in the dataset is our target variable i.e., the dependent variable and ‘Loan_ID’, ‘Gender’, ‘Married’, ‘Dependents’, ‘Education’, ‘Self_Employed’, ‘Property_Area’, ‘Applicant_Income’, ‘Coapplicant_Income’, ‘Loan_Amount’, ‘Loan_Amount_Term’, ‘Credit_History’ are the 12 independent variables.

Based on all the conditions in the dataset we have to determine whether the loan status is approved or not i.e., ‘Yes’ meaning the loan is approved or ‘No’ meaning the opposite of it.

- We provide a brief description of the data with –

```
df.describe()
```

	Dependents	Applicant_Income	Coapplicant_Income	LoanAmount	Loan_Amount_Term	Credit_History
count	956.000000	981.000000	981.000000	954.000000	961.000000	902.000000
mean	0.790795	5179.795107	1601.916330	142.511530	342.201873	0.835920
std	1.038605	5695.104533	2718.772806	77.421743	65.100602	0.370553
min	0.000000	0.000000	0.000000	9.000000	6.000000	0.000000
25%	0.000000	2875.000000	0.000000	100.000000	360.000000	1.000000
50%	0.000000	3800.000000	1110.000000	126.000000	360.000000	1.000000
75%	2.000000	5516.000000	2365.000000	162.000000	360.000000	1.000000
max	3.000000	81000.000000	41667.000000	700.000000	480.000000	1.000000

df.describe() here gives us the five-point summary of the numerical variables and hence a clearer vision of the dataset. Here we see the count, mean, standard deviation, minimum value, first quartile, median, third quartile and maximum value.

DATA PREPROCESSING:

Credit History and **dependents** take up 2 (0, 1) and 4 (0, 1, 2, 3+) values respectively. Since they are taking categorical values, we convert them to object data type as follows –

```
df['Credit_History'] = df['Credit_History'].astype('O')
```

```
df['Dependents'] = df['Dependents'].astype('O')
```

Further describing the categorical and numerical data –

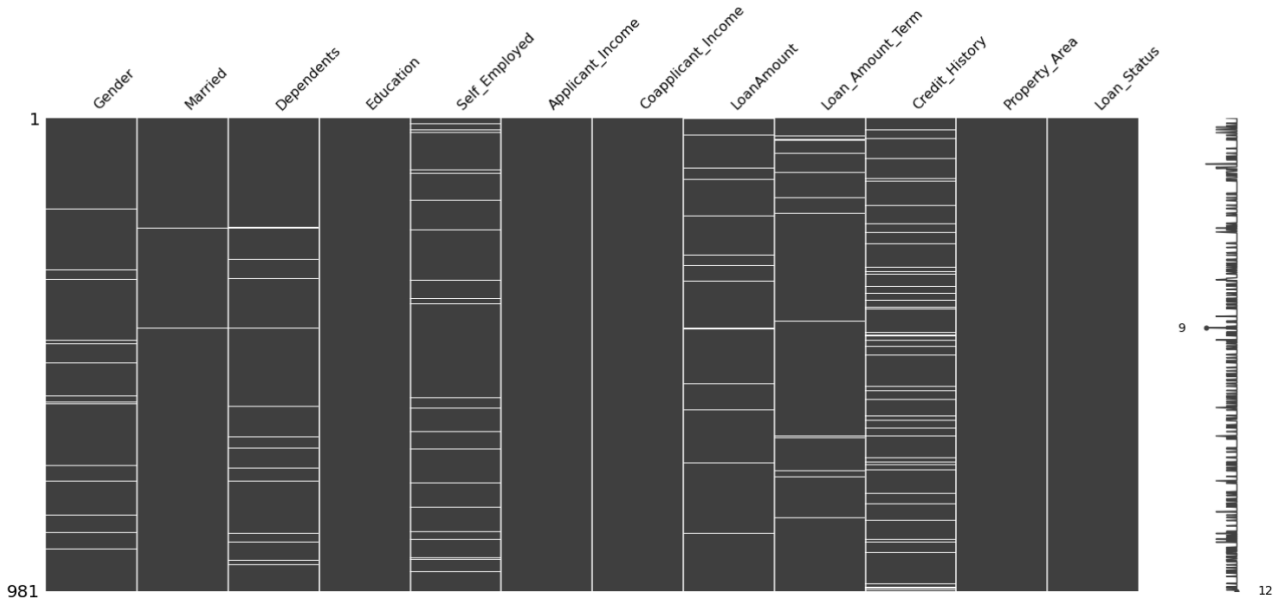
```
df.describe(include='O')
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	Credit_History	Property_Area	Loan_Status
count	981	957	978	956.0	981	926	902.0	981	981
unique	981	2	2	4.0	2	2	2.0	3	2
top	LP002161	Male	Yes	0.0	Graduate	No	1.0	Semiurban	Y
freq	1	775	631	544.0	763	807	754.0	349	721

The ‘Loan_ID’ column is then dropped since it is an indicator variable and wouldn’t serve any purpose in our model building calculations.

- **Missing values:** The data pre-processing begins by finding whether any missing values are present in dataset or not. So, we have used an exploratory visualization to detect the missing values.

```
msno.matrix(df)
plt.show()
```



Further, we check for the number of missing values by isnull() function.

```
df.isnull().sum()
```

```
Gender               24
Married               3
Dependents           25
Education             0
Self_Employed        55
Applicant_Income      0
Coapplicant_Income    0
LoanAmount           27
Loan_Amount_Term     20
Credit_History       79
Property_Area         0
Loan_Status           0
dtype: int64
```

- **Data Cleaning:** We detect the count and location of the null values**.**

First, we check for the numerical variables –

We see ‘LoanAmount’ and ‘Loan_Amount_Term’ have 27 and 20 such values respectively.

To overcome that we will use the Imputation technique to estimate the missing values for the few missing data.

```
df[numerical].isnull().sum()
```

```
Applicant_Income       0
Coapplicant_Income     0
LoanAmount            27
Loan_Amount_Term      20
dtype: int64
```

```
df['LoanAmount'].fillna(df['LoanAmount'].median(),inplace = True)
df['Loan_Amount_Term'].fillna(df['Loan_Amount_Term'].mode()[0],inplace = True)
```

- For the 'LoanAmount' we use median instead of mean for imputation by assuming that since the variable distribution is unknown mean might not exist but it will necessarily have a median.
- For the 'Loan_Amount_Term' we see the value 360 has a larger frequency compared to other values. So here we impute with mode.

We perform the same operation for the Categorical variables –

In case of categorical variables, we use mode for imputation of missing values.

```
df[categorical].isnull().sum()
```

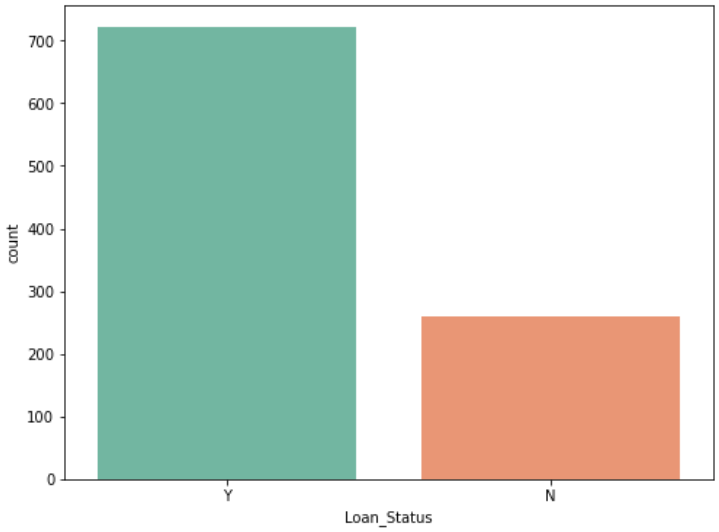
```
Gender            24
Married            3
Dependents        25
Education          0
Self_Employed     55
Credit_History    79
Property_Area      0
Loan_Status        0
dtype: int64
```

```
df['Gender'].fillna(df['Gender'].mode()[0], inplace= True)
df['Married'].fillna(df['Married'].mode()[0], inplace= True)
df['Dependents'].fillna(df['Dependents'].mode()[0], inplace= True)
df['Self_Employed'].fillna(df['Self_Employed'].mode()[0], inplace= True)
df['Credit_History'].fillna(df['Credit_History'].mode()[0], inplace= True)
```

UNIVARIATE VISUAL ANALYSIS:

We will start off with the dependent variable which is our target variable as well. We will analyse this categorical variable using a bar chart as shown below.

Here our target variable is – **Loan status**

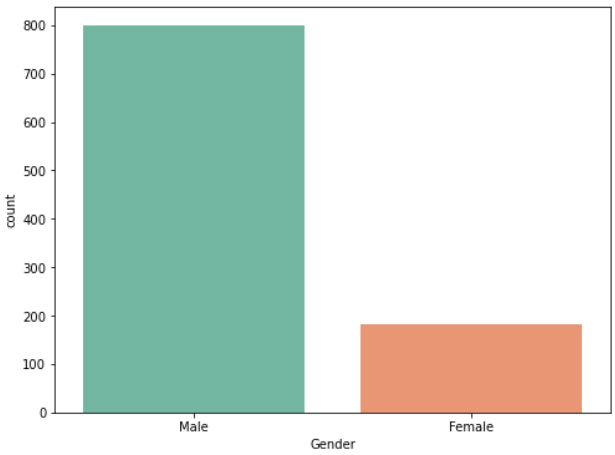


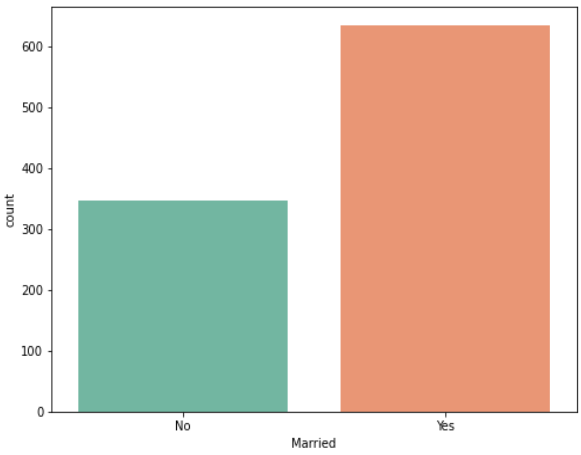
Here, from the bar chart we can see that loan of 721 people out of 981 people was approved i.e., loan of approximately 73% people was approved.

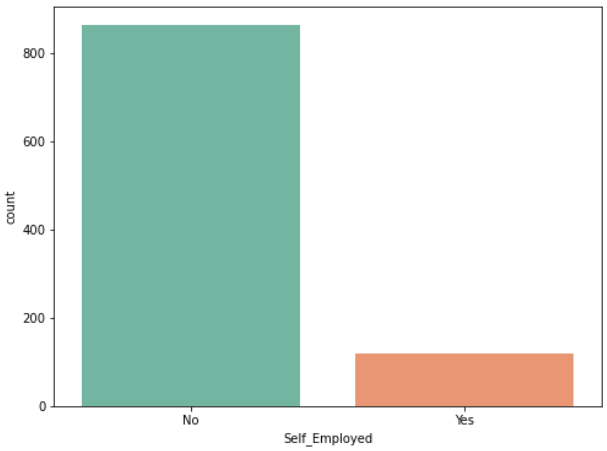
There are 2 types of Independent Variables: **Categorical & Numerical**.

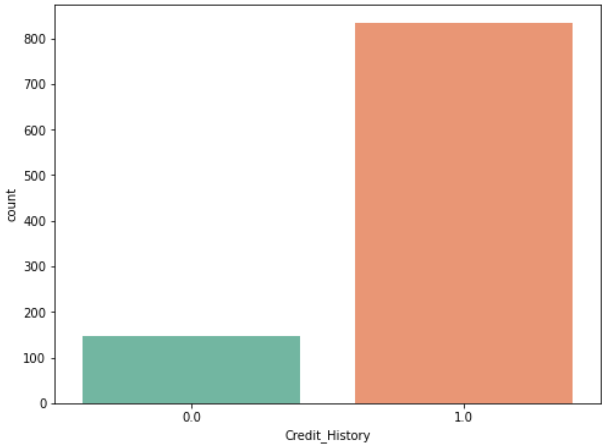
Exploring other categorical variables

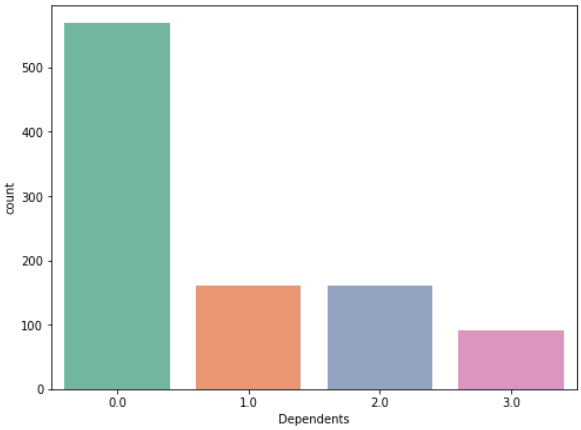
Categorical Features are Gender, Marital Status, Employment Type, and Credit History, Number of Dependents, Education Level, Property or Area Background.

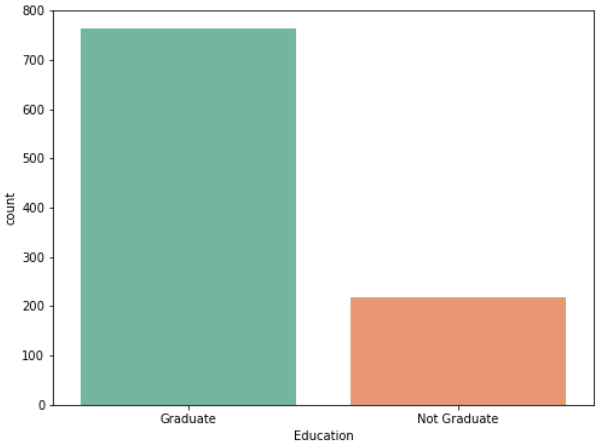


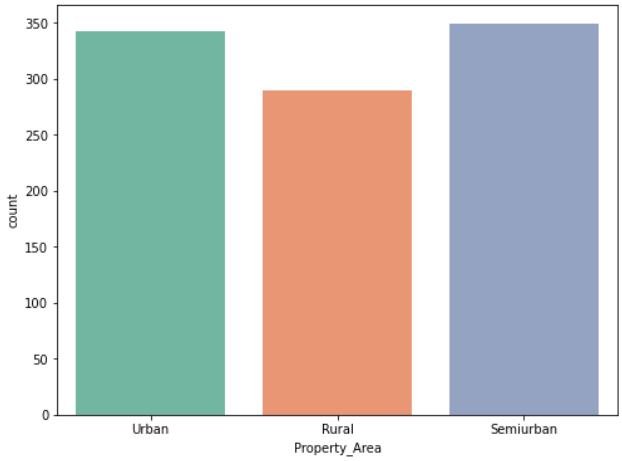










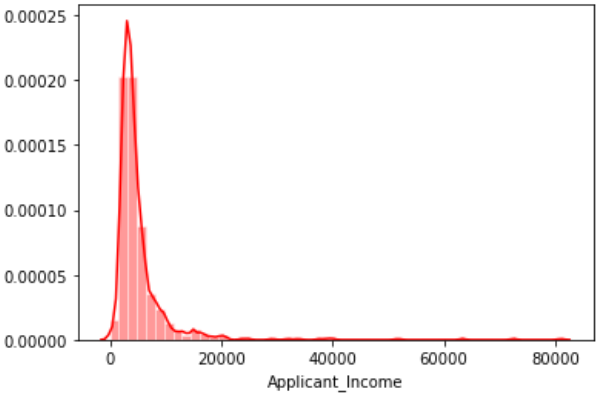


From the above bar plots, we can say that in our data set:

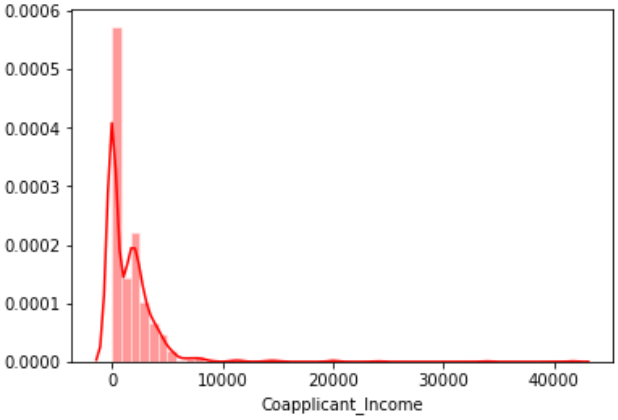
- 81% loan applicants are male in our data set.
- Nearly 64% are married.
- Nearly 87% - 89% are self-employed.
- Loan has been approved for more than approximately 70% of the applicants.
- Approximately 60% of the applicants have no dependents.
- More than 70% applicants are graduates.
- Highest number of applicants are from semi urban area followed by urban and rural area.

Exploring numerical variables

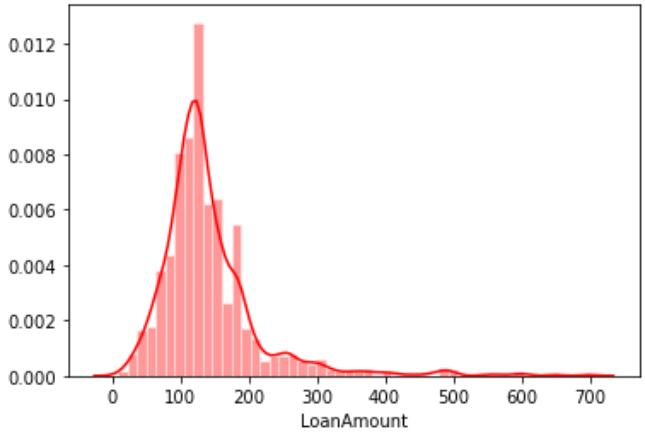
Numerical features are Applicant`s income, Co-applicant`s income, Loan amount, Loan amount term.



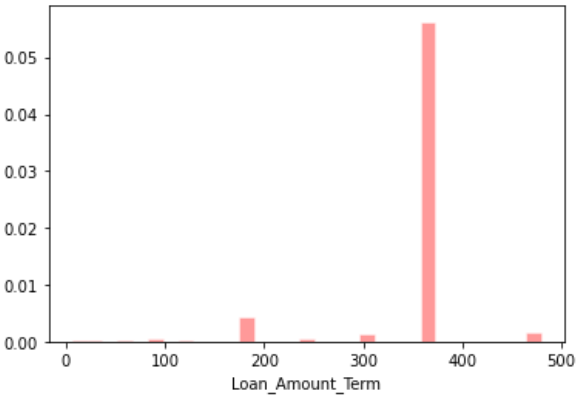
It can be inferred that most of the data in Applicant Income are towards the left which means it is not normally distributed i.e., data in Applicant`s income is skewed.



It can be inferred from the graph that most of the co-applicant`s income is towards the left which implies that data in co-applicant`s income is skewed.

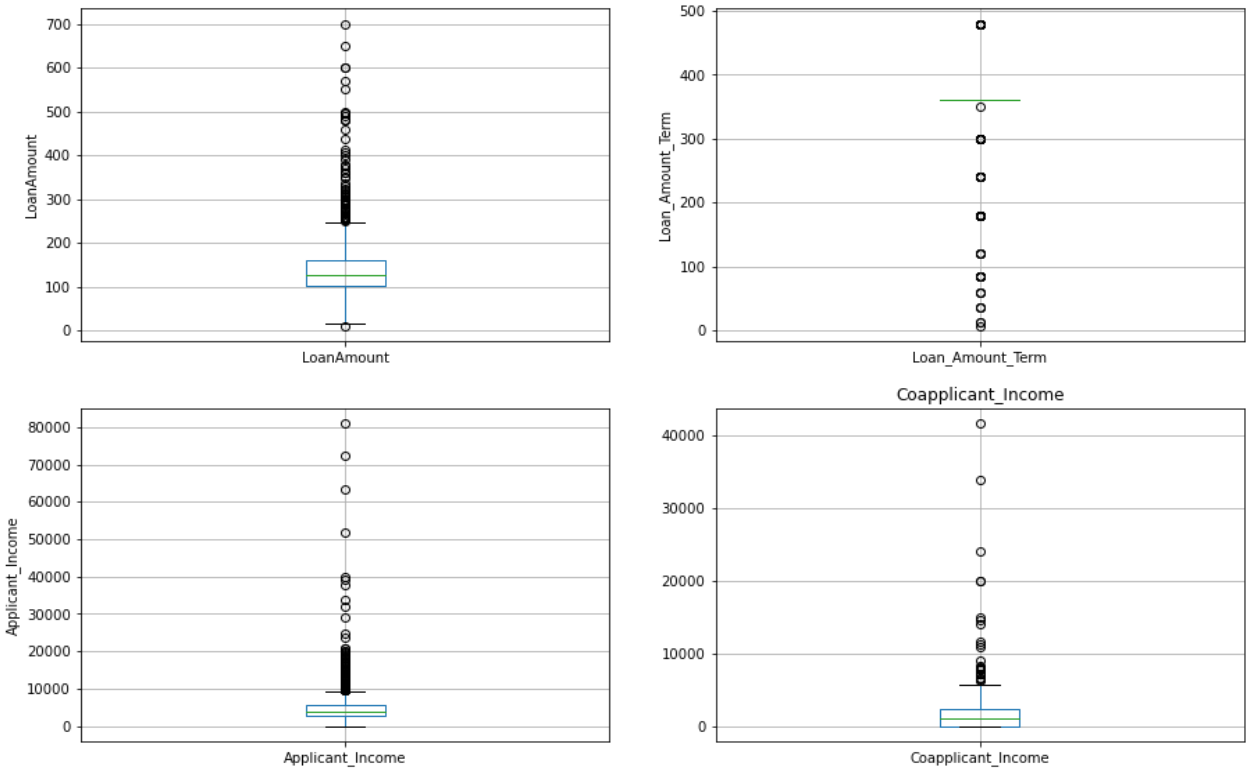


It can be seen from the graph that the data in Loan amount is very slightly skewed.



It can be seen from the graph that the data in Loan amount term are discrete and most of the data lies in between 350 – 400.

- **Outlier detection:** We need to find if any outliers are present in the dataset.



We can infer from the above boxplots that there are some outlies in the data of the numerical variables. Now we will try to remove It. Since the variables are skewed, we will use Interquartile range technique to find and remove the outliers.

By using IQR method we see that,

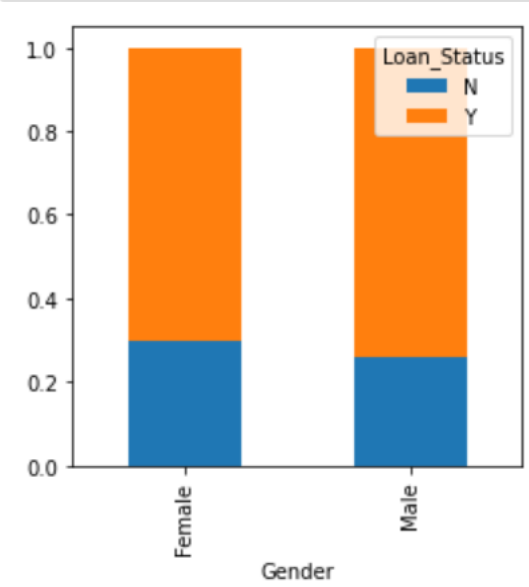
- Applicant Income outliers are values < -4534.5 or > 12682.0
- Co-applicant Income outliers are values < -7103.25 or > 9471.0
- Loan amount outliers are values < -76.0 or > 337.0

The above boxplot says that Applicant income is approximately in between 5000 – 20000. It also says that the Co-applicant income is lesser than applicant income and is within 5000 – 15000.

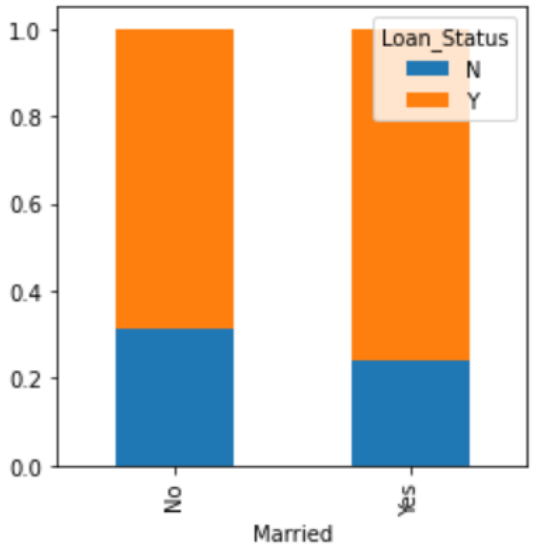
BIVARIATE VISUAL ANALYSIS:

Bivariate analysis is an analysis of two variables to determine the relationships between them. Specifically, the dependent vs independent Variables.

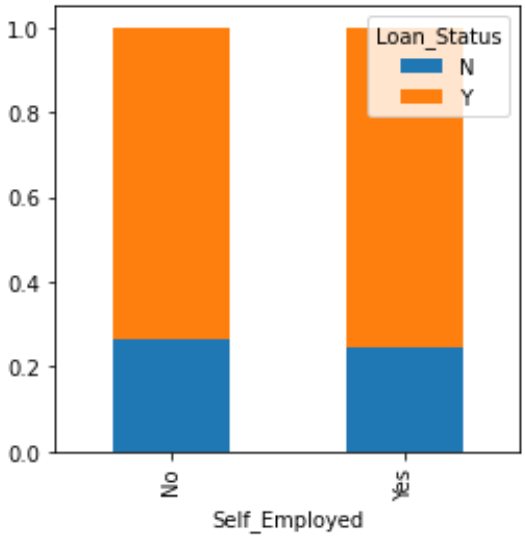
Categorical Independent variable vs Target variable



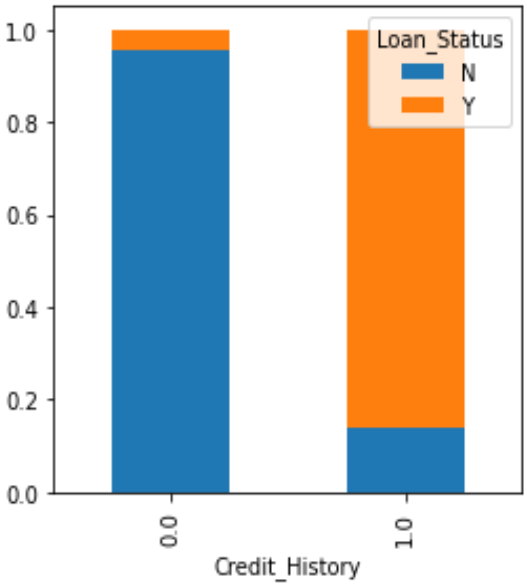
It can be inferred that the proportion of male and female applicants is more or less the same for both approved and unapproved loans.



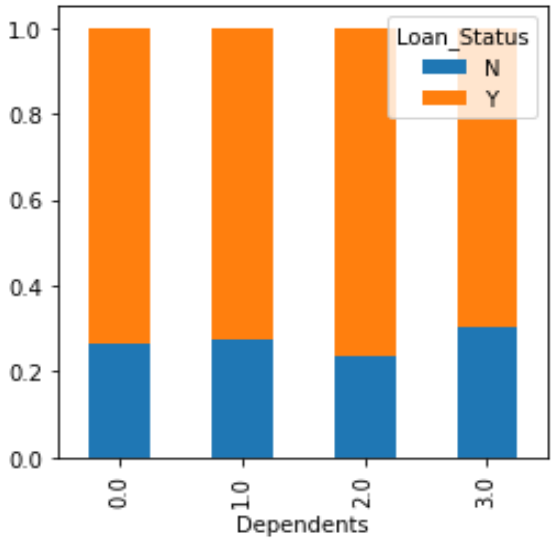
Married applicants have higher chance of loan approval.



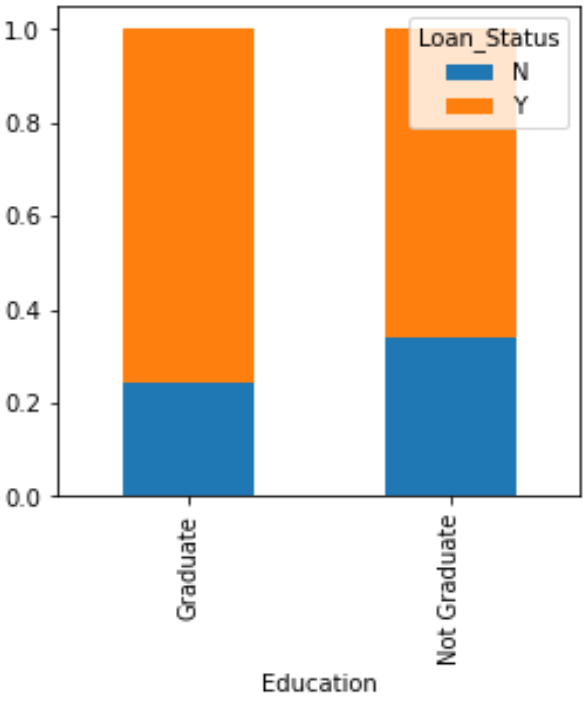
There is nothing significant we can infer from Self-employed vs Loan Status plot.



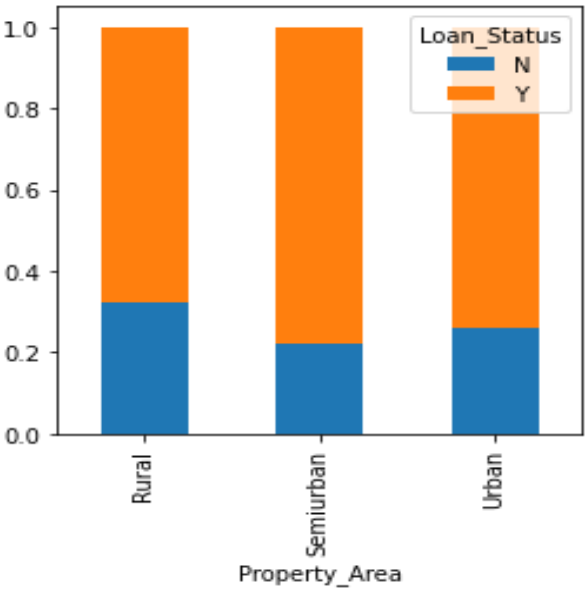
It seems people with a credit history of 1 are more likely to get their loans approved.



The distribution of applicants with 1 or 3+ dependents is similar across both the categories of Loan Status.



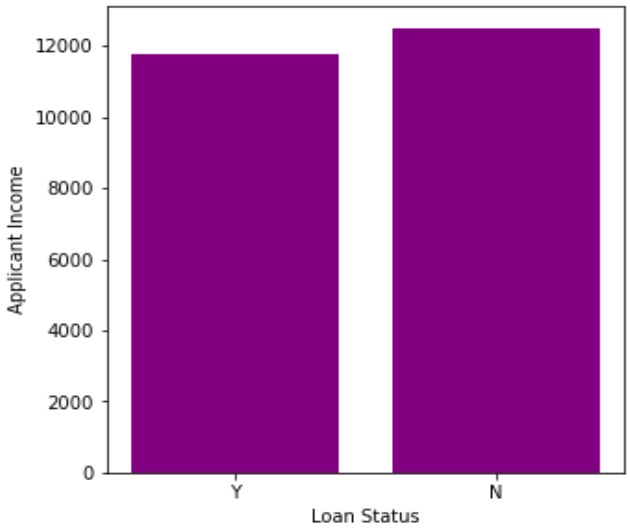
Graduate applicants have higher chance of loan approval.



The proportion of loans getting approved in semi-urban areas is higher as compared to that in rural or urban areas.

Numerical Independent variable vs Target variable

We tried to find the mean income of people for which the loan has been approved vs the mean income of people for which the loan has not been approved but we don't see any changes in the mean income.



So, we make bins for the applicant income variable based on the values like:

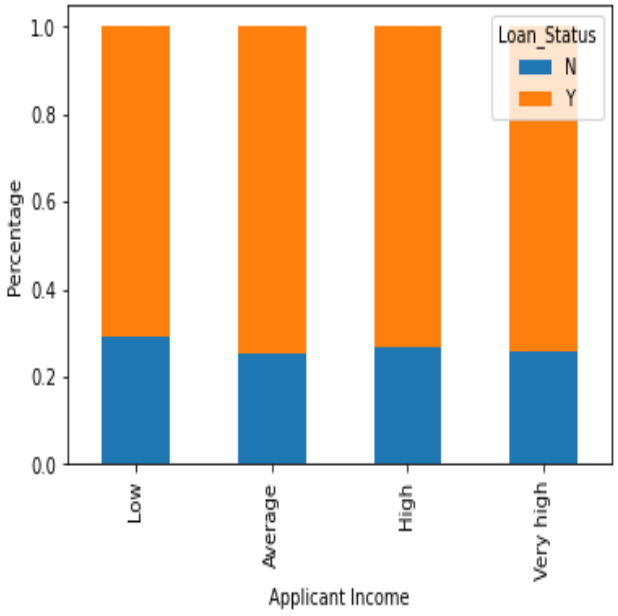
Low: 0-2500

Average: 2500-5000

High: 5000-7500

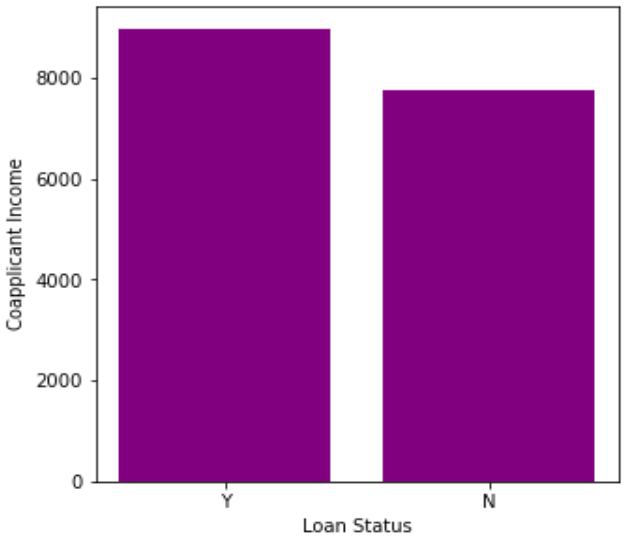
Very High: 7500-81000

and analyse the corresponding loan status for each bin.



It can be inferred that Applicant's income does not affect the chances of loan approval which contradicts our hypothesis in which we assumed that if the applicant's income is high the chances of loan approval will also be high

Next, we will analyse the Co-applicant's Income and Loan Amount variable in a similar manner. Here the y-axis represents the mean Co-applicant Income. We don't see any change in the mean income. So, let's make bins for the Co-applicant Income variable based on the values in it and analyse the corresponding loan status for each bin.



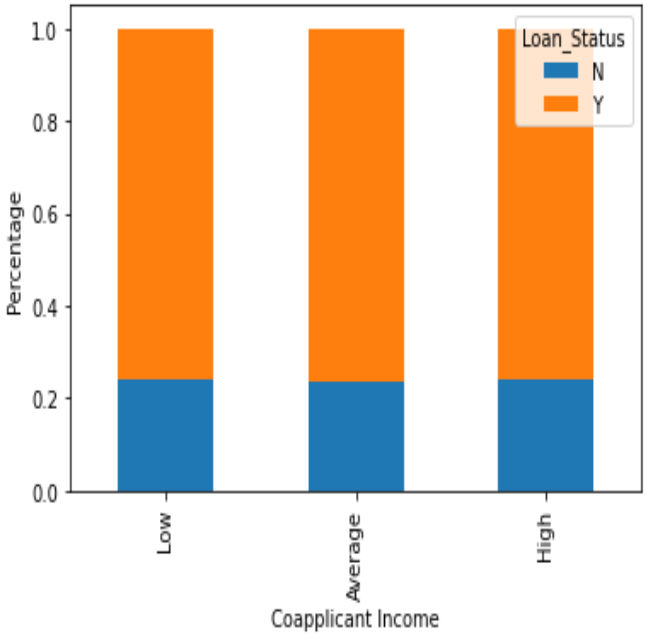
So, we make bins for the Co-applicant Income variable based on the values like:

Low: 0-2000

Average: 2000-4000

High: 4000-42000

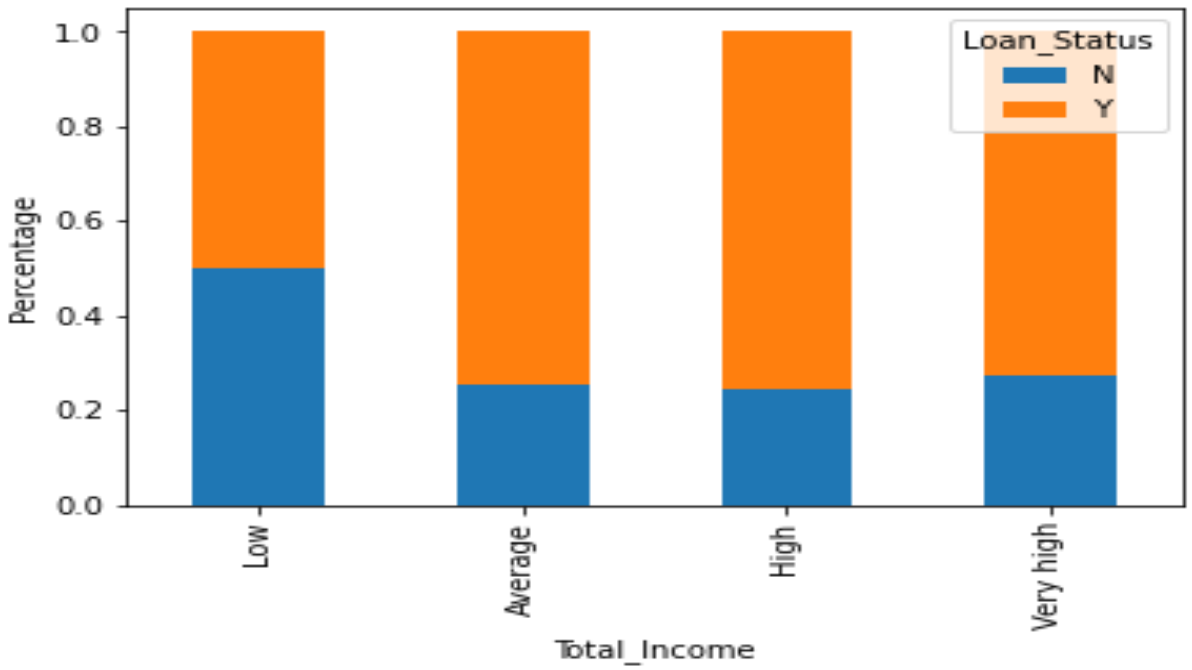
and analyse the corresponding loan status for each bin.



It shows that if Co-applicant's Income is less the chances of loan approval are high.

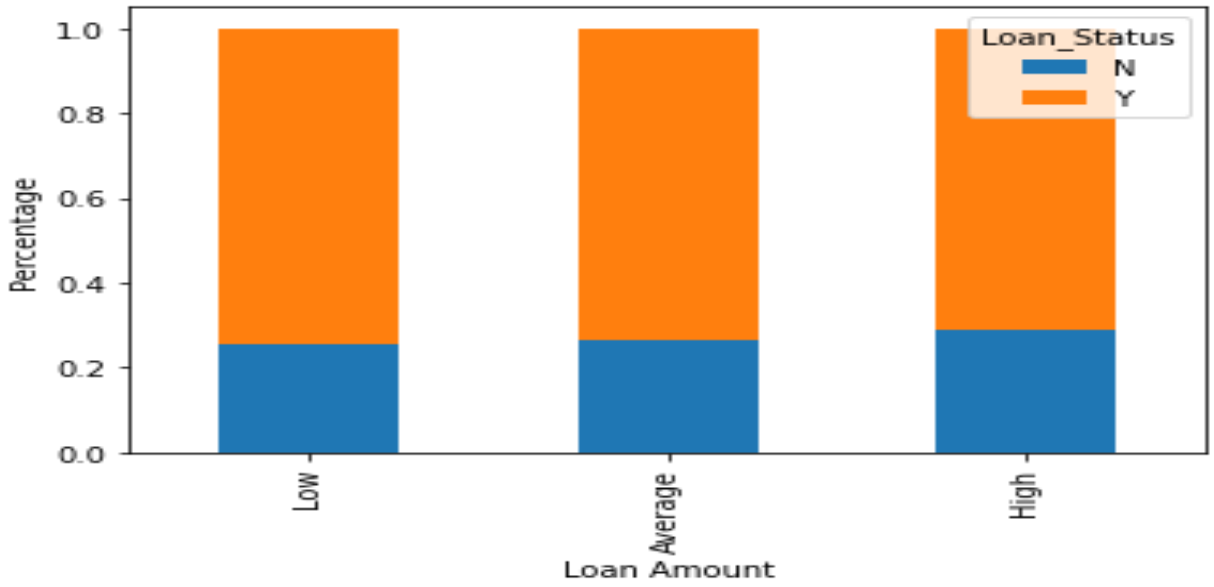
But this does not look right. The possible reason behind this may be that most of the applicants don't have any co-applicant so the co-applicant income for such applicants is 0 and hence the loan approval is not dependent on it. So, we can make a new variable in which we will combine the Applicant's and Co-applicants' Income to visualize the combined effect of income on loan approval.

Let us combine the Applicant Income and Co-applicant Income and see the combined effect of Total Income on the Loan Status.



We can see that Proportion of loans getting approved for applicants having low Total Income is very less as compared to that of applicants with Average, High, and Very High Income.

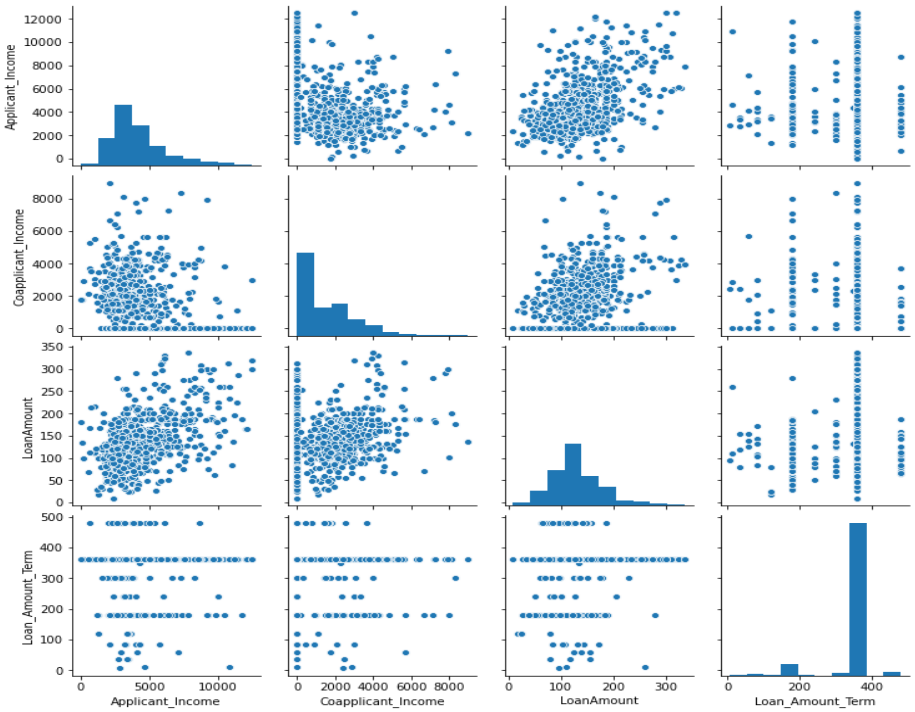
Now, let's visualize the Loan amount variable.



It can be seen that the proportion of approved loans is higher for Low and Average Loan Amounts as compared to that of High Loan Amounts which considered that the chances of loan approval will be high when the loan amount is less.

MULTIVARIATE VISUAL ANALYSIS:

A **pair plot** allows us to see both distribution of single variables and relationships between two variables. The pairs plot builds on two basic figures, the histogram and the scatter plot.



From this pair plot we can see that data in the applicant income and co-applicant income column are skewed whereas the data in the loan amount term is showing discrete pattern.

A **correlation heatmap** is a graphical representation of a correlation matrix. Correlation heatmaps are a type of plot that visualize the strength of relationships between numerical variables. Correlation plots are used to understand which variables are related to each other and the strength of this relationship. The color-coding of the cells makes it easy to identify relationships between variables at a glance. Correlation heatmaps can be used to find both linear and nonlinear relationships between variables.

Multicollinearity is the occurrence of high intercorrelations among two or more independent variables in a multiple regression model.

We can say that from this heatmap, in this data no significant multicollinearity is present.

Encoding Categorical Data

Encoding is the process of converting the data or a given sequence of characters, symbols, alphabets etc., into a specified format, for the secured transmission of data.

Here we will encode the independent categorical variable for building the model by using get_dummies() function. Since, Loan status is the dependent or target variable we will drop that column.

MODEL BUILDING:

After completing the exploratory data analysis, the next step is to build the model. The quality of data will have a huge impact on the quality of the model. In this section, firstly the covariates and the target variables are declared.

It is implemented in python by applying the following codes:

```
X = dff
y = df4['Loan_Status']
```

Here, the target variable is 'Loan_Status', which is a qualitative data.

Splitting the dataset

Secondly, it is also required to split the dataset into test data and train data. The train data denotes the subset of a dataset used for training the machine learning model based on the outputs in the dataset. The test data is the subset of the dataset, used for testing the machine learning model. The Machine Learning model predicts the outcomes of the test dataset. Here, the dataset is divided into 70-30 ratio and 70% of the data are considered as the train data and remaining 30% data are considered as the test data. This is because the larger part is for training purposes and the smaller part is for testing purposes. This is more important because using the same data for training and testing would not produce good results.

It is implemented in python by applying the following codes:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 0)
```

Output:

In the dataset, the number of individuals in the train data and the test data are 642 and 272 respectively.

Feature Scaling

The final step in this phase is feature scaling. The learning algorithms perform much better when the features are on the same scale. A well-prepared data always gives better results. So, it is important to fine-tune the data at every step of building the model. Here, normalization technique is used to scaling the dataset. It is a method to rescale the features of the data within a specific range mostly [0,1]. To normalize the data min-max scaling method is applied to each feature column. The min-max scaler can be written as-

$$X_{new} = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$

It is implemented in python by applying the following codes:

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)
```

Model Training

This is the stage where the ML algorithm is trained by feeding datasets. This is the stage where the learning takes place. Consistent training can significantly improve the prediction rate of the ML model. The weights of the model must be initialized randomly. This way the algorithm will learn to adjust the weights accordingly. The following machine learning models are used in this section:

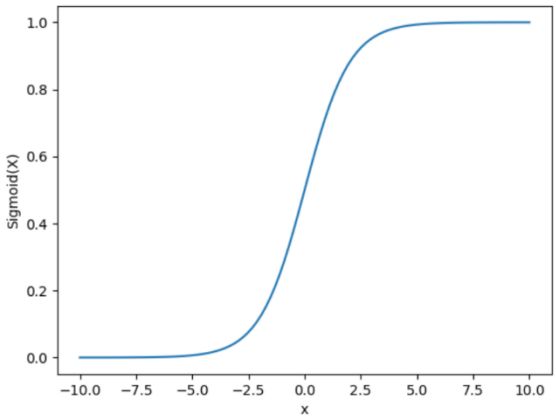
1. Logistic Regression
2. Decision Tree
3. Random Forest
4. K-Nearest Neighbor
5. Naïve Bayes Classifier

Logistic Regression

Logistic regression is a statistical analysis method to predict a binary outcome, such as yes or no, based on prior observations of a data set. A logistic regression model predicts a dependent data variable by analyzing the relationship between one or more existing independent variables. In this example, it is required to come to a decision if the person is eligible for the loan or not. These binary outcomes allow straightforward decisions between two alternatives. A Logistic Regression model is similar to a Linear Regression model, except that the Logistic Regression utilizes a more sophisticated cost function, which is known as the "Sigmoid function" or "logistic function" instead of a linear function. It can be expressed as:

$$\log\left(\frac{p(X)}{1-p(X)}\right) = \beta_0 + \sum_{i=1}^{n} \beta_i x_i$$

In order to map predicted values to probabilities, we use the Sigmoid function. The function maps any real value into another value between 0 and 1. Graphically, a sigmoid function can be represented as:



The logistic regression is implemented in python by applying the following codes:

```
from sklearn.linear_model import LogisticRegression
LR = LogisticRegression()
LR.fit(X_train, y_train)
```

Model Outputs and Interpretations:

The following outputs are obtained from the model:

Confusion Matrix:

The following confusion matrix is obtained from the model:

```
[[ 33  37]
 [  2 204]]
```

Classification Report:

The following classification report is obtained from the model:

	precision	recall	f1-score	support
N	0.94	0.47	0.63	70
Y	0.85	0.99	0.91	206
accuracy			0.86	276
macro avg	0.89	0.73	0.77	276
weighted avg	0.87	0.86	0.84	276

Accuracy Score:

The Logistic Regression Model is 85.87% Accurate.

K-Fold Cross-Validation

To determine if our model is overfitting or not, we need to test it on the Validation set. If a given model does not perform well on the validation set then it's going to perform worse when dealing with real live data. This notion makes Cross-Validation probably one of the most important concepts of machine learning which ensures the stability of our model. Cross-Validation is just a method that simply reserves a part of data from the dataset and uses it for testing the model (Validation set), and the remaining data other than the reserved one is used to train the model.

The K-Fold Cross-Validation is implemented in python by applying the following codes:

```
from sklearn.model_selection import cross_val_score

scores = cross_val_score(LR, X_train, y_train, cv = 10, scoring='accuracy')

print('Cross-validation scores:{}'.format(scores))
```

Model Outputs and Interpretations:

The average cross validation score is obtained by applying the following codes:

```
print('Average cross-validation score:{:.4f}'.format(scores.mean()))
```

```
Average cross-validation score:0.8832
```

We performed a binary classification using Logistic regression as our model and cross-validated it using 10-Fold cross-validation. The average accuracy of our model after using cross validation is approximately 88.32%. The classification report model reported an accuracy score of 85.87%.

Decision Tree

A decision tree is a supervised machine learning algorithm mainly used for Regression and Classification. It breaks down a data set into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. A decision tree can handle both categorical and numerical data.

The decision tree is implemented in python by applying the following codes:

```
from sklearn.tree import DecisionTreeClassifier
DT = DecisionTreeClassifier(criterion = 'entropy', random_state = 100)
DT.fit(X_train,y_train)
```

In this model, the criterion 'entropy' is considered for information gain. The criterion function is used to measure the quality of a split. There are also some other criterions such as 'gini' and 'log_loss' etc. And, the function 'random_state' controls the randomness of the estimator. The features are always randomly permuted at each split.

Model Outputs and Interpretations:

The following outputs are obtained from the model:

Confusion Matrix:

The following confusion matrix is obtained from the model:

```
[[ 36  34]
 [ 24 182]]
```

Classification Report:

The following classification report is obtained from the model:

	precision	recall	f1-score	support
N	0.60	0.51	0.55	70
Y	0.84	0.88	0.86	206
accuracy			0.79	276
macro avg	0.72	0.70	0.71	276
weighted avg	0.78	0.79	0.78	276

Accuracy Score:

The Decision Tree Model is 78.99% Accurate.

Random Forest

It is a type of ensemble learning method, where a group of weak models combine to form a powerful model. The random forest starts with a standard machine learning technique called a "decision tree" which, in ensemble terms, corresponds to our weak learner. In Random Forest, we grow multiple trees as opposed to a single tree. To classify a new object based on attributes, each tree gives a classification. The forest chooses the classification having the most votes (Over all the trees in the forest) and in case of regression, it takes the average of outputs by different trees.

The random forest is implemented in python by applying the following codes:

```
from sklearn.ensemble import RandomForestClassifier
RF = RandomForestClassifier(random_state = 123)
RF.fit(X_train, y_train)
```

Model Outputs and Interpretations:

The following outputs are obtained from the model:

Confusion Matrix:

The following confusion matrix is obtained from the model:

```
[[ 33  37]
 [  4 202]]
```

Classification Report:

The following classification report is obtained from the model:

	precision	recall	f1-score	support
N	0.89	0.47	0.62	70
Y	0.85	0.98	0.91	206
accuracy			0.85	276
macro avg	0.87	0.73	0.76	276
weighted avg	0.86	0.85	0.83	276

Accuracy Score:

The Random Forest Model is 85.14% accurate.

K-Nearest Neighbor

K-Nearest Neighbor is one of the simplest Machine Learning algorithms based on Supervised Learning technique. It assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data. It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.

The K-Nearest Neighbor is implemented in python by applying the following codes:

```
from sklearn.neighbors import KNeighborsClassifier
KNN = KNeighborsClassifier(n_neighbors = 3)
KNN.fit(X_train, y_train)
```

In this model, ‘n_neighbors’ denotes Number of neighbors to use by default for k-neighbors queries. Here we have taken the number of neighbors to be 3.

Model Outputs and Interpretations:

The following outputs are obtained from the model:

Confusion Matrix:

The following confusion matrix is obtained from the model:

```
[[ 32  38]
 [  8 198]]
```

Classification Report:

The following classification report is obtained from the model:

	precision	recall	f1-score	support
N	0.80	0.46	0.58	70
Y	0.84	0.96	0.90	206
accuracy			0.83	276
macro avg	0.82	0.71	0.74	276
weighted avg	0.83	0.83	0.82	276

Accuracy Score:

The K-Nearest Neighbor Model is 83.33% Accurate.

Naive Bayes Classifier

Naïve Bayes algorithm is a supervised learning algorithm, which is based on Bayes theorem and used for solving classification problems. Bayes' theorem is used to determine the probability of a hypothesis with prior knowledge. It depends on the conditional probability. The formula for Bayes' theorem is given as:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Where, P(A|B) is Posterior probability: Probability of hypothesis A on the observed event B. P(B|A) is Likelihood probability: Probability of the evidence given that the probability of a hypothesis is true. P(A) is Prior Probability: Probability of hypothesis before observing the evidence and P(B) is Marginal Probability: Probability of Evidence. It is a probabilistic classifier, which means it predicts on the basis of the probability of an object.

The Naive Bayes algorithm is implemented in python by applying the following codes:

```
from sklearn.naive_bayes import GaussianNB
NBC=GaussianNB()
NBC.fit(X_train,y_train)
```

Model Outputs and Interpretations:

The following outputs are obtained from the model:

Confusion Matrix:

The following confusion matrix is obtained from the model:

```
[[ 33  37]
 [  3 203]]
```

Classification Report:

The following classification report is obtained from the model.

	precision	recall	f1-score	support
N	0.92	0.47	0.62	70
Y	0.85	0.99	0.91	206
accuracy			0.86	276
macro avg	0.88	0.73	0.77	276
weighted avg	0.86	0.86	0.84	276

Accuracy Score:

The Naive Bayes classifier Model is 85.51% accurate.

Model Comparison:

The following table shows the accuracy of all the models that are fitted using the dataset:

	Models	Accuracy
1	Logistic Regression	0.883245
2	Random Forest	0.851449
3	Decision Tree	0.789855
4	K Nearest Neighbor	0.833333
5	Naive Bayes Classifier	0.855072

Based on the models that has been fitted, summarised in the table, model that obtain the highest accuracy to predict the model is Logistic Regression. The Decision Tree model is the least accurate model compared to the others.

CONCLUSION

To predict whether the loan should be approved or not, we collected the data from Kaggle, which contains 980 rows of 13 features. By performing exploratory data analysis and building necessary classification models, we predict the Loan Status of the applicants. From the predictive analysis we conclude the following:

- The original Logistic Regression model shows an accuracy of 85.87% (approx.). After performing cross validation, the model accuracy becomes 88.32% (average). Hence, we can conclude cross validation improves the model performance in this case.
- The Random Forest model provides us with an accuracy of 85.14%.
- The K Nearest Neighbor model gives us an accuracy of 83% approximately.
- The Naïve Bayes Classifier model shows an accuracy of 85.5% which is also pretty good.
- The Decision Tree model gives an accuracy of 79% (approx.) which is the least among all the model performances. Hence in this case, Decision Tree performs least accurately.
- Amazingly Logistic Regression worked better than other Ensembled models. In case our dataset, Logistic Regression gives us the best result among all. Hence, we can conclude that this model does the best job of predicting whether the loan will be approved or not.

However, we reached a pretty good accuracy, but it's not the end. There are still many places of improvements by tuning the hyper parameters or by using more complex models. We can further try different models with different subset of features. Machine Learning and Data Science are a huge ocean of opportunities to try and gamble out various techniques and processes. We can try to use a bigger dataset to get a more accurate idea about the dataset. More data always increase the precision of the algorithm. However, this needs official support from banks to face the real-life data problems.

REFERENCE

- Dileep B. Desai, Dr. R.V.Kulkarni “A Review: Application of Data Mining Tools in CRM for Selected Banks”, (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 4 (2), 2013, 199 –201.M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.
- X.Frencis Jensy, V.P.Sumathi,Janani Shiva Shri-An exploratory Data Analysis for Loan Prediction based on nature of clients, International Journal of Recent Technology and Engineering (IJRTE), Volume-7 Issue-4S, November 2018.
- G. Arutjothi, Dr. C. Senthamarai- Prediction of Loan Status in Commercial Bank using Machine Learning Classifier, proceedings of the International Conference on Intelligent.
- Anchal Goyal, Ranpreet Kaur- A survey on ensemble model of Loan Prediction, International journal of engineering trends and application (IJETA), Vol. 3 Issue 1, Jan-Feb 2016.
- J.H. Aboobyda, and M.A. Tarig, “Developing Prediction Model of Loan Risk in Banks Using Data Mining”, Machine Learning and Applications: An International Journal (MLAIJ), vol. 3, no.1, pp. 1–9, 2016. K. Elissa, "Title of paper if known," unpublished.
- Dataset Source: Kaggle