MINI PROJECT REPORT

On

Loan Approval Prediction

Submitted by

Name of Student: Ankita Kushwaha

Roll No: 171500050

Name of Student: Haimangi Sahgal

Roll No: 171500117

Department of Computer Engineering & Applications

Institute of Engineering & Applications

GLA UNIVERSITY
MATHURA-281406, INDIA

2019

ACKNOWLEDGEMENT

The success and final outcome of this project required a lot of guidance and assistance from many people and I am extremely privileged to have got this all along the completion of my project. All that I have done is only due to such supervision and assistance and I would not forget to thank them.

I respect and thank **Mr. Pankaj Sharma**, for providing me an opportunity to do the project work in **GLA University** and giving me all support and guidance which made me complete the project duly. I am extremely thankful to him for providing such a nice support and guidance, although he had busy schedule managing the corporate affairs.

I owe my deep gratitude to our project guide **Mr. Pankaj Sharma** who took keen interest on my project work and guided me all along, till the completion of our project work by providing all the necessary information for developing a good project.

I would like to thank the lab staff for the operation extended to us throughout the project. After doing this project I can confidently say that this experience has not only enriched me with technical knowledge but also has unparsed the maturity of thought and vision. The attributes required in being a successful professional

Ankita Kushwaha

Haimangi Sahgal

Declaration

I hereby declare that the work which is being presented in the Mini Project "**Loan Approval Prediction"**, is an authentic record of my own work carried under the supervision of **Mr. Pankaj Sharma.**

Signature of Candidates:

Name of Candidates: Ankita Kushwaha

Haimangi Sahgal

Roll. No : 171500050

171500117

Course: CSE

Year: III

Semester-V

ABSTRACT

In today's world there are many risks involved in bank loans, so as to reduce their capital loss; banks should perform the risk and assessment analysis of the individual before sanctioning loan. In the absence of this process there are many chances that this loan may turn in to bad loan in near future. The bank employees are not able to analyse or predict whether the customer can payback the amount or not (good customer or bad customer) for the given interest rate Banks hold huge volumes of customer behavior related data from which they are unable to arrive at a decision point i.e. if an applicant can be defaulter or not. This model can be used by the organizations in making the right decision to approve or reject the loan request of the customers. In India, the number of people applying for the loans gets increased for various reasons in recent years. The aim of this paper is to find the nature of the client applying for the personal loan. An exploratory data analysis technique is used to deal with this problem. The analysis result is short term loans are preferred by majority of the clients and the clients majorly apply loans for debt consolidation.

Contents

CHAPTER-1

INTRODUCTION

Distribution of the loans is the core business part of almost every banks. The main portion the bank's assets is directly came from the profit earned from the loans distributed by the banks. The prime objective in banking environment is to invest their assets in safe hands where it is. Today many banks/financial companies approves loan after a regress process of verification and validation but still there is no surety whether the chosen applicant is the deserving right applicant out of all applicants. Through this system we can predict whether that particular applicant is safe or not and the whole process of validation of features is automated by machine learning technique. The disadvantage of this model is that it emphasize different weights to each factor but in real life sometime loan can be approved on the basis of single strong factor only, which is not possible through this system.

The Loan Prediction System can can automatically calculate the weight of each features taking part in loan processing and on new test data same features are processed with respect to their associated weight .A time limit can be set for the applicant to check whether his/her loan can be sanctioned or not. Loan Prediction System allows jumping to specific application so that it can be check on priority basis. This Paper is exclusively for the managing authority of Bank/finance company, whole process of prediction is done privately no stakeholders would be able to alter the processing. Result against particular Loan Id can be send to various department of banks so that they can take appropriate action on application. This helps all others department to carried out other formalities.

1.1 Problem Statement

1.1.1 Business Problem

"Dream Housing Finance company deals in all home loans. They have presence across all urban, semi urban and rural areas. Customer first apply for home loan after that company validates the customer eligibility for loan. Company wants to automate the loan eligibility process (real time) based on customer detail provided while filling online application form. These details are

Gender, Marital Status, Education, Number of Dependents, Income, Loan Amount, Credit History and others. To automate this process, they have given a problem to identify the customers segments, those are eligible for loan amount so that they can specifically target these customers."

Loan prediction is a very common real-life problem that every retail bank faces in their lending operations. If the loan approval process is automated, it can save a lot of man hours and improve the speed of service to the customers. The increase in customer satisfaction and savings in operational costs are significant. However, the benefits can only be reaped if the bank has a robust model to accurately predict which customer's loan it should approve and which to reject, in order to minimize the risk of loan default.

1.1.2 Translate Business Problem into Data Science / Machine Learning problem

This is a classification problem where we have to predict whether a loan will be approved or not. Specifically, it is a binary classification problem where we have to predict either one of the two classes given i.e. approved (Y) or not approved (N). Another way to frame the problem is to predict whether the loan will likely to default or not, if it is likely to default, then the loan would not be approved, and vice versa. The dependent variable or target variable is the Loan_Status, while the rest are independent variable or features. We need to develop a model using the features to predict the target variable.

1.2 Motivation

The loan is one of the most important products of the banking. All the banks are trying to figure out effective business strategies to persuade customers to apply their loans. However, there are some customers behave negatively after their application are approved. To prevent this situation, banks have to find some methods to predict customers’ behaviours. Machine learning algorithms have a pretty good performance on this purpose, which are widely-used by the banking. Here, I will work on loan behaviours prediction using machine learning models.

1.3 Objective

From a proper analysis of positive points and constraints on the component, it can be safely concluded that the product is a highly efficient component. The aim of this Paper is to provide quick, immediate and easy way to choose the deserving applicants. It can provide special advantages to the bank. There have been numbers cases of computer glitches, errors in content and most important weight of features is fixed in automated prediction system, So in the near future the so –called software could be made more secure, reliable and dynamic weight adjustment .In near future this module of prediction can be integrate with the module of automated processing system

CHAPTER-2

HYPOTHESIS GENERATION

Hypothesis Generation is the process of listing out all the possible factors that can affect the outcome i.e. which of the features will have an impact on whether a loan will be approved or not. Some of the hypothesis are:

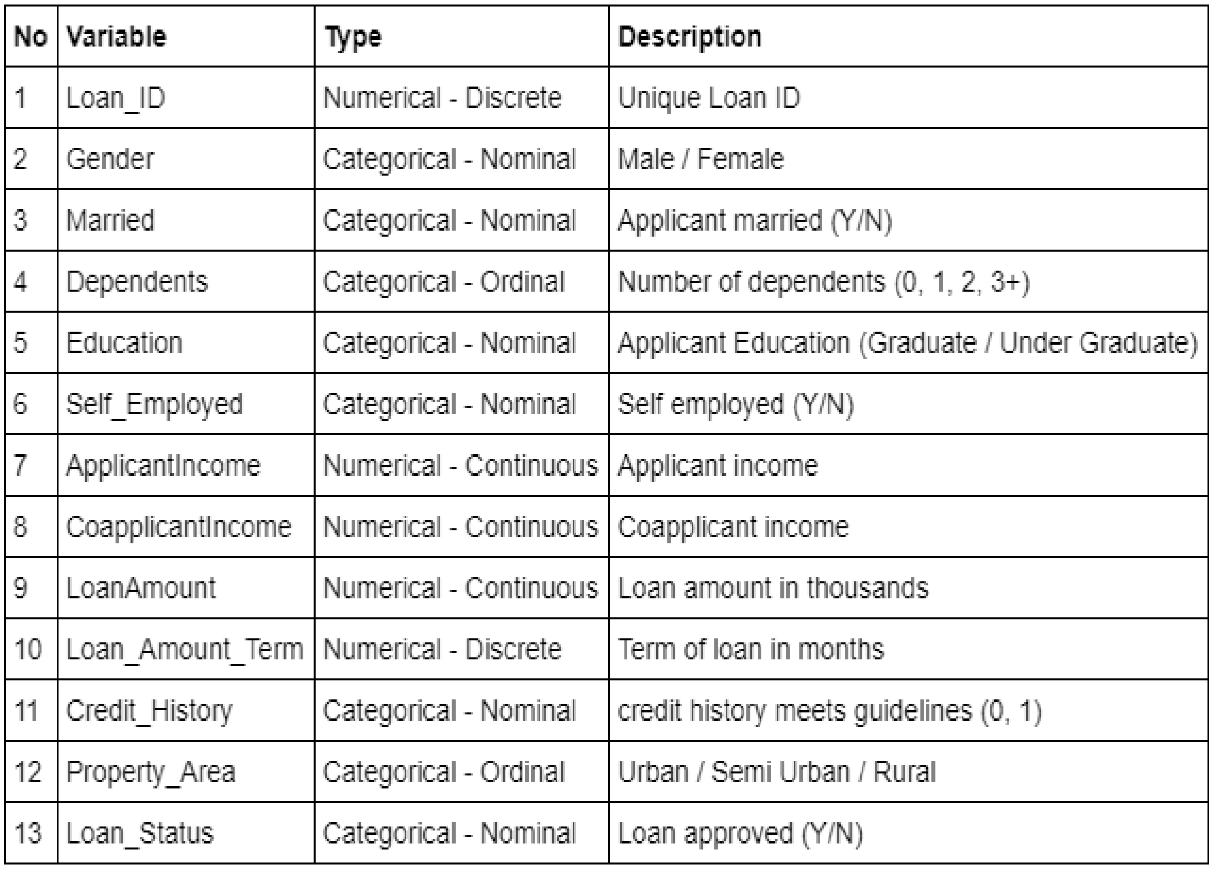
1. Education - Applicants with higher education level i.e. graduate level should have higher chances of loan approval
2. Income: Applicants with higher income should have more chances of loan approval
3. Loan amount: If the loan amount is less, the chances of loan approval should be high
4. Loan term: Loans with shorter time period should have higher chances of approval
5. Previous credit history: Applicants who have repayed their previous debts should have higher chances of loan approval
6. Monthly installment amount: If the monthly installment amount is low, the chances of loan approval should be high
7. And so on.

Some of the hypothesis seem intuitive while others may not. We will try to validate each of these hypothesis based on the dataset.

CHAPTER-3

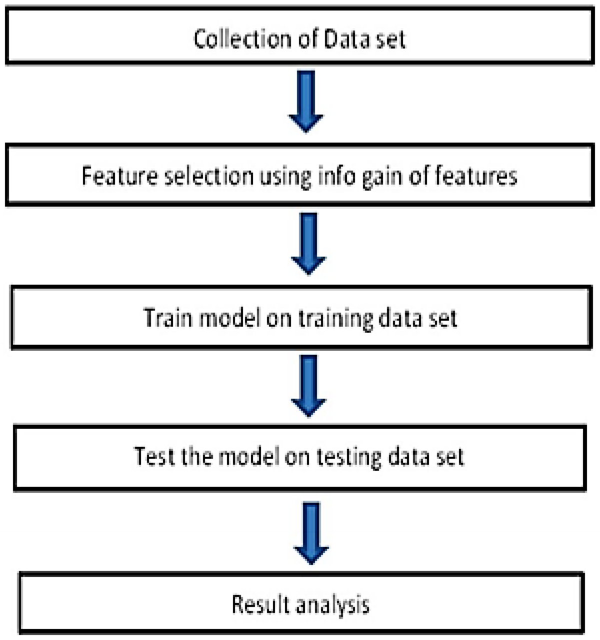
DATA COLLECTION

The training data set is now supplied to machine learning model, on the basis of this data set the model is trained. Every new applicant details filled at the time of application form acts as a test data set. After the operation of testing, model predict whether the new applicant is a fit case for approval of the loan or not based upon the inference it conclude on the basis of the training data sets.



No	Variable	Type	Description
1	Loan_ID	Numerical - Discrete	Unique Loan ID
2	Gender	Categorical - Nominal	Male / Female
3	Married	Categorical - Nominal	Applicant married (Y/N)
4	Dependents	Categorical - Ordinal	Number of dependents (0, 1, 2, 3+)
5	Education	Categorical - Nominal	Applicant Education (Graduate / Under Graduate)
6	Self_Employed	Categorical - Nominal	Self employed (Y/N)
7	ApplicantIncome	Numerical - Continuous	Applicant income
8	CoapplicantIncome	Numerical - Continuous	Coapplicant income
9	LoanAmount	Numerical - Continuous	Loan amount in thousands
10	Loan_Amount_Term	Numerical - Discrete	Term of loan in months
11	Credit_History	Categorical - Nominal	credit history meets guidelines (0, 1)
12	Property_Area	Categorical - Ordinal	Urban / Semi Urban / Rural
13	Loan_Status	Categorical - Nominal	Loan approved (Y/N)

3.1 Loan Prediction Methodology


Collection of Data set
Feature selection using info gain of features
Train model on training data set
Test the model on testing data set
Result analysis

CHAPTER-4

STATISTICAL ANALYSIS

We will use Python to explore the data in order to gain a better understanding of the features and target variable. We will also analyze the data to summarize their main characteristics.

Note: There are 3 data types in the data

1. object: Object format means variables are categorical. Categorical variables in our dataset are: Loan_ID, Gender, Married, Dependents, Education, Self_Employed, Property_Area, Loan_Status
2. int64: It represents the integer variables. ApplicantIncome is of this format.
3. float64: It represents the variable which have some decimal values involved. They are also numerical variables. Numerical variables in our dataset are: CoapplicantIncome, LoanAmount, Loan_Amount_Term, and Credit_History

4.1 Importance of Data Types

Datatypes are an important concept because statistical methods can only be used with certain data types. You have to analyze continuous data differently than categorical data otherwise it would result in a wrong analysis. Therefore knowing the types of data you are dealing with, enables you to choose the correct method of analysis.

4.2 Types of Variables

4.2.1 Independent Variables

An independent variable is the variable that is changed or controlled in a scientific experiment to test the effects on the dependent variable. The independent variables, also known in a statistical context as regressors, represent inputs or causes, that is, potential reasons for variation. In experiment, any variable that the experimenter manipulates can be called an independent variable.

4.2.2 Dependent Variable

A dependent variable is the variable being tested and measured in a scientific experiment. The dependent variable is 'dependent' on the independent variable. As the experimenter changes the independent variable, the effect on the dependent variable is observed and recorded.

4.2.3 Categorical Variable

In statistics, a categorical variable is a variable that can take on one of a limited, and usually fixed number of possible values, assigning each individual or other unit of observation to a particular group or nominal category on the basis of some qualitative property. Commonly (though not in this article), each of the possible values of a categorical variable is referred to as a level. The probability distribution associated with a random categorical variable is called a categorical distribution.

Categorical variables contain a finite number of categories or distinct groups. Categorical data might not have a logical order. For example, categorical predictors include gender, material type, and payment method.

Categorical variables can be further categorised into ordinal and nominal variables.

4.2.3.1 Ordinal categorical variable

A categorical variable whose categories can be meaningfully ordered is called ordinal. For example, a student's grade in an exam (A, B, C or Fail) is ordinal.

4.2.3.2 Nominal categorical variable

It does not matter which way the categories are ordered in tabular or graphical displays of the data -- all orderings are equally meaningful.

4.2.4 Target Variable

The "target variable" is the variable whose values are to be modeled and predicted by other variables. It is analogous to the dependent variable (i.e., the variable on the left of the equal sign) in linear regression. There must be one and only one target variable in a decision tree analysis.

4.2.5 Numerical Variable

A numerical variable is a variable where the measurement or number has a numerical meaning. They can be further classified into discrete and continuous variables.

4.2.5.1 Discrete numerical variable

A variable whose values are whole numbers (counts) is called discrete. For example, the number of items bought by a customer in a supermarket is discrete.

4.2.5.2 Continuous numerical variable

A variable that may contain any value within some range is called continuous. For example, the time that the customer spends in the supermarket is continuous.

4.2 Univariate analysis

Univariate is an expression, equation, function or polynomial of only one variable. Univariate analysis applies to data sets that consist of a single variable. If the data set consists of continuous variables, then the measures of interest are the central tendency and spread of the variable. For categorical features we can use frequency table or bar plots which will calculate the number of each category in a particular variable. For numerical features, a histogram or a box-plot can be used to look at the distribution of the variable. With a histogram, you can check the central tendency, variability, modality, and kurtosis of a distribution. Note that a histogram can't show you if you have any outliers. This is why we also use box-plots.

4.3 Bivariate Analysis

Bivariate analysis is one of the simplest forms of quantitive (statistical) analysis.It involves the analysis of two variables (often denoted as *X*, *Y*), for the purpose of determining the empirical relationship between them.

Bivariate analysis can be helpful in testing simple hypotheses of association. Bivariate analysis can help determine to what extent it becomes easier to know and predict a value for one variable (possibly a dependent_variable) if we know the value of the other variable (possibly the independent variable).

Bivariate analysis can be contrasted with univariate analysis in which only one variable is analysed. Like univariate analysis, bivariate analysis can be descriptive or inferential. It is the analysis of the relationship between the two variables.

CHAPTER-5

DATA PREPROCESSING

Data pre-processing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. Data pre-processing is a method of resolving such issues.

5.1 Missing value and outlier treatment

After exploring all the variables in our data, we can now impute the missing values and treat the outliers because missing data and outliers can have adverse effect on the model performance.

5.1.1 Missing Value Imputation

In statistics, imputation is the process of replacing missing data with substituted values. When substituting for a data point, it is known as "unit imputation"; when substituting for a component of a data point, it is known as "item imputation". There are three main problems that missing data causes: missing data can introduce a substantial amount of bias, make the handling and analysis of the data more arduous, and create reductions in efficiency, missing data can create problems for analyzing data, imputation is seen as a way to avoid pitfalls involved with listwise deletion of cases that have missing values.

5.1.2 Outlier Treatment

An outlier is a data point that differs significantly from other observations. An outlier may be due to variability in the measurement or it may indicate experimental error; the latter are sometimes excluded from the data set.

LoanAmount contains outliers so we have to treat them as the presence of outliers affects the distribution of the data. Having outliers in the dataset often has a significant effect on the mean and standard deviation and hence affecting the distribution. We must take steps to remove outliers from our data sets.

Due to these outliers bulk of the data in the loan amount is at the left and the right tail is longer. This is called right skewness (or positive skewness). One way to remove the skewness is by doing the log transformation. As we take the log transformation, it does not affect the smaller values much, but reduces the larger values. So, we get a distribution similar to normal distribution.

CHAPTER-6

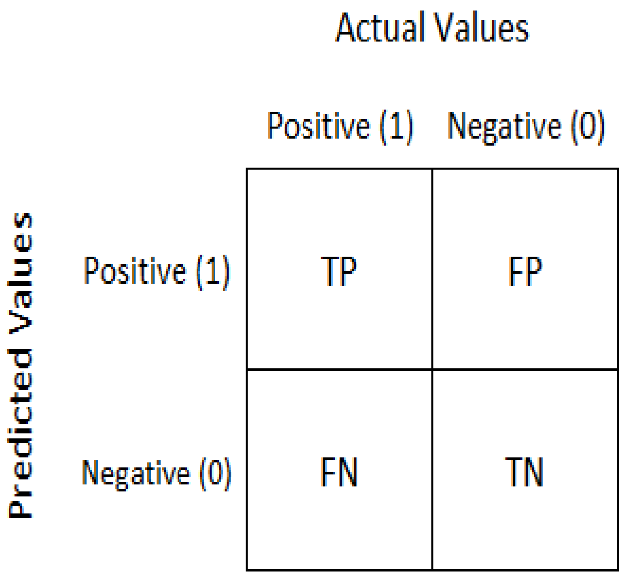
MODEL DEVELOPMENT AND EVALUATION

6.1 Evaluation Metrics for Classification Problems

The process of model building is not complete without evaluation of model's performance. We can plot the results and compare them with the actual values, i.e. calculate the distance between the predictions and actual values. 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:

6.1.1 Accuracy

It using the confusion matrix which is a tabular representation of Actual vs Predicted values. This is how a confusion matrix looks like:



True Positive – Targets which are actually true(Y) and we have predicted them true(Y)

True Negative – Targets which are actually false(N) and we have predicted them false(N)

False Positive – Targets which are actually false(N) but we have predicted them true(T)
False Negative – Targets which are actually true(T) but we have predicted them false(N)

Using these values, we can calculate the accuracy of the model. The accuracy is given by:
Accuracy = (TP+TN) / (TP+TN+FP+FN)

6.1.2 Precision

It is a measure of correctness achieved in true prediction i.e. of observations labeled as true, how many are actually labeled true.
Precision = TP / (TP + FP)

6.1.3 Recall (Sensitivity)

It is a measure of actual observations which are predicted correctly i.e. how many observations of true class are labeled correctly. It is also known as ‘Sensitivity’. E.g. Proportion of patients with a disease who test positive.
Recall = TP / (TP + FN)

6.1.4 Specificity

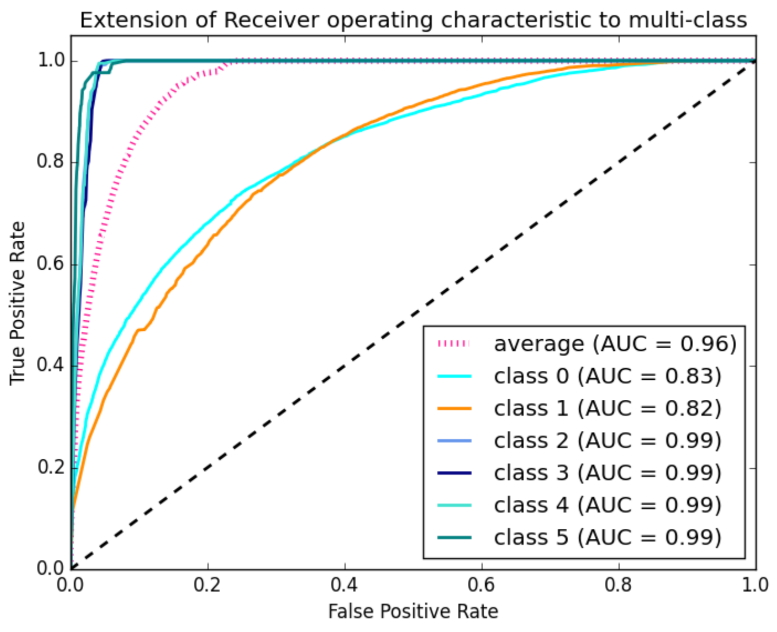
It is a measure of how many observations of false class are labeled correctly. E.g. Proportion of patients without the disease who test negative.
Specificity = TN / (TN + FP)
Specificity and Sensitivity plays a crucial role in deriving ROC curve.

6.1.5 ROC curve

1. Receiver Operating Characteristic(ROC) summarizes the model’s performance by evaluating the trade offs between true positive rate (Sensitivity) and false positive rate (1- Specificity).
2. The area under curve (AUC), referred to as index of accuracy(A) or concordance index, is a perfect performance metric for ROC curve. Higher the area under curve, better the prediction power of the model.

3. The area of this curve measures the ability of the model to correctly classify true positives and true negatives. We want our model to predict the true classes as true and false classes as false.
4. So it can be said that we want the true positive rate to be 1. But we are not concerned with the true positive rate only but the false positive rate too. For example in our problem, we are not only concerned about predicting the Y classes as Y but we also want N classes to be predicted as N.
5. We want to increase the area of the curve which will be maximum for class 2,3,4 and 5 in the above example.
6. For class 1 when the false positive rate is 0.2, the true positive rate is around 0.6. But for class 2 the true positive rate is 1 at the same false positive rate. So, the AUC for class 2 will be much more as compared to the AUC for class 1. So, the model for class 2 will be better.
7. The class 2,3,4 and 5 model will predict more accurately as compared to the class 0 and 1 model as the AUC is more for those classes.

This is how a ROC curve looks like:



At the competition's page, it has been mentioned that our submission data would be evaluated based on the accuracy. Hence, we will use accuracy as our evaluation metric.

6.2 Model Building : Part I

Our first model to predict the target variable. We will start with Logistic Regression which is used for predicting binary outcome.

1. Logistic Regression is a classification algorithm. It is used to predict a binary outcome (1 / 0, Yes / No, True / False) given a set of independent variables.
2. Logistic regression is an estimation of Logit function. Logit function is simply a log of odds in favor of the event.
3. This function creates a s-shaped curve with the probability estimate, which is very similar to the required step wise function

6.2.1 Logistic Regression using stratified k-folds cross validation

To check how robust our model is to unseen data, we can use Validation. It is a technique which involves reserving a particular sample of a dataset on which you do not train the model. Later, you test your model on this sample before finalizing it. Some of the common methods for validation are listed below:

1. The validation set approach
2. k-fold cross validation
3. Leave one out cross validation (LOOCV)
4. Stratified k-fold cross validation

In this section we will learn about stratified k-fold cross validation. Let us understand how it works:

1. Stratification is the process of rearranging the data so as to ensure that each fold is a good representative of the whole.
2. For example, in a binary classification problem where each class comprises of 50% of the data, it is best to arrange the data such that in every fold, each class comprises of about half the instances.
3. It is generally a better approach when dealing with both bias and variance.
4. A randomly selected fold might not adequately represent the minor class, particularly in cases where there is a huge class imbalance.

5. If K=N, then it is called Leave one out cross validation, where N is the number of observations.

Below is the visualization of a stratified k-fold validation when k=5.

6.3 Feature Engineering

Based on the domain knowledge, we can come up with new features that might affect the target variable. We will create the following three new features:

1. **Total Income** - As discussed during bivariate analysis we will combine the Applicant Income and Coapplicant Income. If the total income is high, chances of loan approval might also be high.
2. **Equated Monthly Installment** - EMI is the monthly amount to be paid by the applicant to repay the loan. Idea behind making this variable is that people who have high EMI's might find it difficult to pay back the loan. We can calculate the EMI by taking the ratio of loan amount with respect to loan amount term.
3. **Balance Income** - This is the income left after the EMI has been paid. Idea behind creating this variable is that if this value is high, the chances are high that a person will repay the loan and hence increasing the chances of loan approval.

6.4 Model Building : Part II

After creating new features, we can continue the model building process. So we will start with logistic regression model and then move over to more complex models like RandomForest and XGBoost.

6.4.1 Logistic Regression

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression(or logit regression) is estimating the parameters of a logistic model (a form of binary regression). Mathematically, a binary logistic model has a dependent variable with two possible values, such as pass/fail which is represented by an indicator variable, where the two values are labeled "0" and "1". In the logistic model, the log-odds (the logarithm of the odds) for the value labeled "1" is a linear combination of one or more independent variables ("predictors"); the independent variables can each be a binary variable (two classes, coded by an indicator variable) or a continuous variable (any real value).

6.4.2 Decision Tree

Decision tree is a type of supervised learning algorithm(having a pre-defined target variable) that is mostly used in classification problems. In this technique, we split the population or sample into two or more homogeneous sets(or sub-populations) based on most significant splitter / differentiator in input variables.

Decision trees use multiple algorithms to decide to split a node in two or more sub-nodes. The creation of sub-nodes increases the homogeneity of resultant sub-nodes. In other words, we can say that purity of the node increases with respect to the target variable.

6.4.3 Random Forest

1. RandomForest is a tree based bootstrapping algorithm wherein a certain no. of weak learners (decision trees) are combined to make a powerful prediction model.
2. For every individual learner, a random sample of rows and a few randomly chosen variables are used to build a decision tree model.
3. Final prediction can be a function of all the predictions made by the individual learners.
4. In case of regression problem, the final prediction can be mean of all the predictions.

There are some parameters worth exploring with the sklearn RandomForestClassifier:

1. n_estimators
2. max_features

n_estimators = ususally bigger the forest the better, there is small chance of overfitting here. The more estimators you give it, the better it will do. We will use the default value of 10.

6.4.4 GridSearchCV

To improve the accuracy by tuning the hyperparameters for this model. Use grid search to get the optimized values of hyper parameters. GridSearch is a way to select the best of a family of hyper parameters, parametrized by a grid of parameters.

We will use GridSearchCV in sklearn.model_selection for an exhaustive search over specified parameter values for an estimator. GridSearchCV implements a "fit" and a "score" method. It also implements "predict", "predict_proba", "decision_function", "transform" and "inverse_transform" if they are implemented in the estimator used.

We will tune the max_depth and n_estimators parameters. max_depth decides the maximum depth of the tree and n_estimators decides the number of trees that will be used in random forest model.

6.4.5 XGBoost

XGBoost is a fast and efficient algorithm and has been used to by the winners of many data science competitions.

XGBoost works only with numeric variables and we have already replaced the categorical variables with numeric variables. Let's have a look at the parameters that we are going to use in our model.

1. n_estimator: This specifies the number of trees for the model.
2. max_depth: We can specify maximum depth of a tree using this parameter.

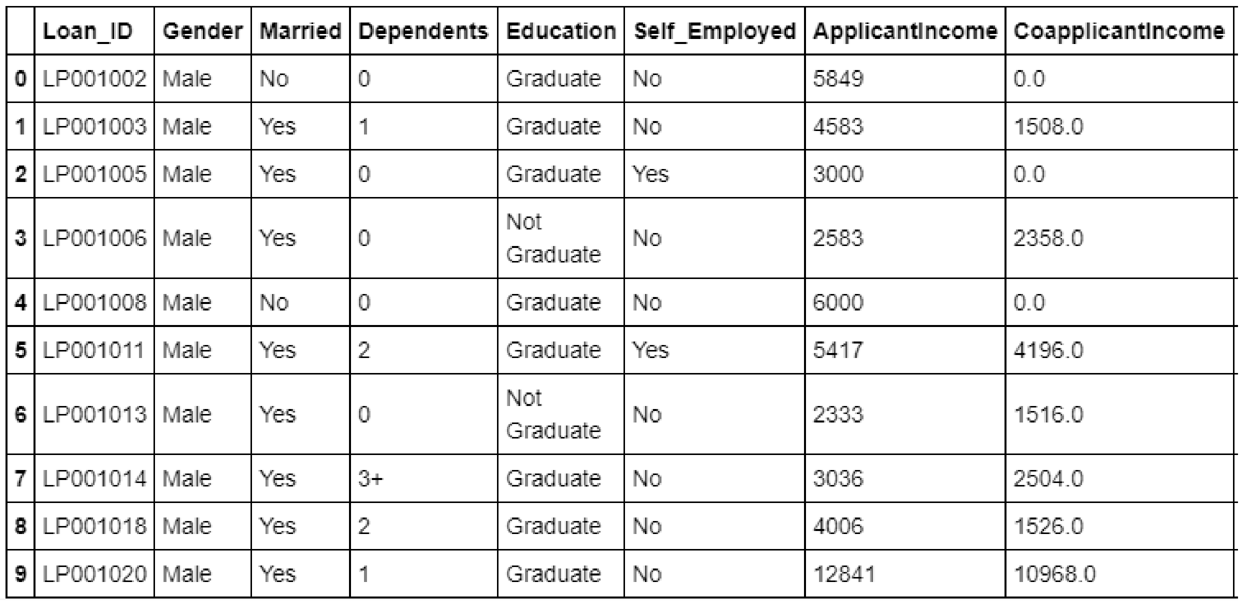
CHAPTER-7

EXPLORATING DATA ANALYSIS

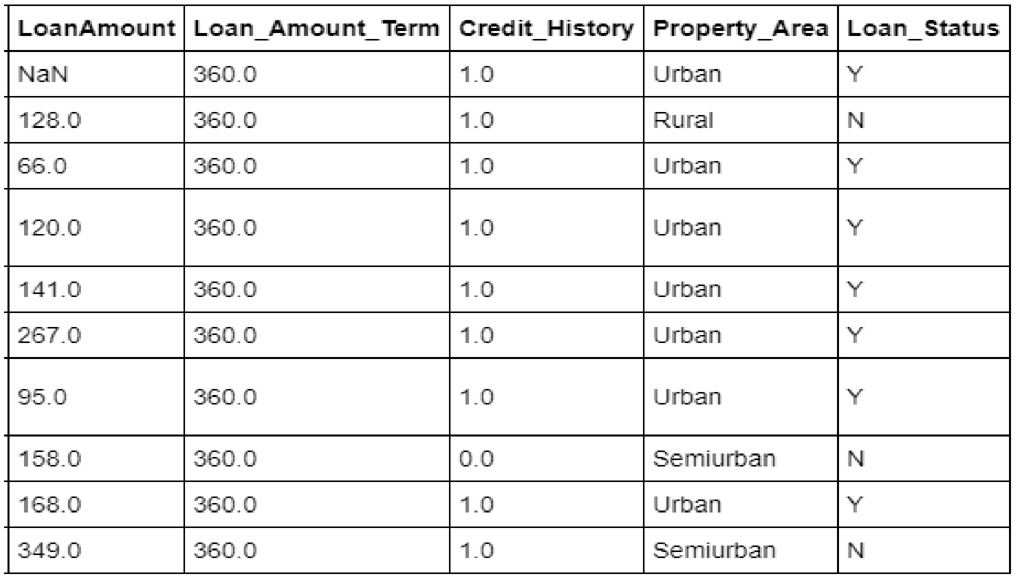
In [1]:
```
# Importing Library
import pandas as pd
import numpy as np
from sklearn import preprocessing
from sklearn.preprocessing import LabelEncoder
# Reading the training dataset in a dataframe using Pandas
df = pd.read_csv("train.csv")
# Reading the test dataset in a dataframe using Pandas
test = pd.read_csv("test.csv")
```

In [2]:
```
# First 10 Rows of training Dataset
df.head(10)
```

Out[2]:



	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome
0	LP001002	Male	No	0	Graduate	No	5849	0.0
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0
4	LP001008	Male	No	0	Graduate	No	6000	0.0
5	LP001011	Male	Yes	2	Graduate	Yes	5417	4196.0
6	LP001013	Male	Yes	0	Not Graduate	No	2333	1516.0
7	LP001014	Male	Yes	3+	Graduate	No	3036	2504.0
8	LP001018	Male	Yes	2	Graduate	No	4006	1526.0
9	LP001020	Male	Yes	1	Graduate	No	12841	10968.0



LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
NaN	360.0	1.0	Urban	Y
128.0	360.0	1.0	Rural	N
66.0	360.0	1.0	Urban	Y
120.0	360.0	1.0	Urban	Y
141.0	360.0	1.0	Urban	Y
267.0	360.0	1.0	Urban	Y
95.0	360.0	1.0	Urban	Y
158.0	360.0	0.0	Semiurban	N
168.0	360.0	1.0	Urban	Y
349.0	360.0	1.0	Semiurban	N

In [3]: # Store total number of observation in training dataset

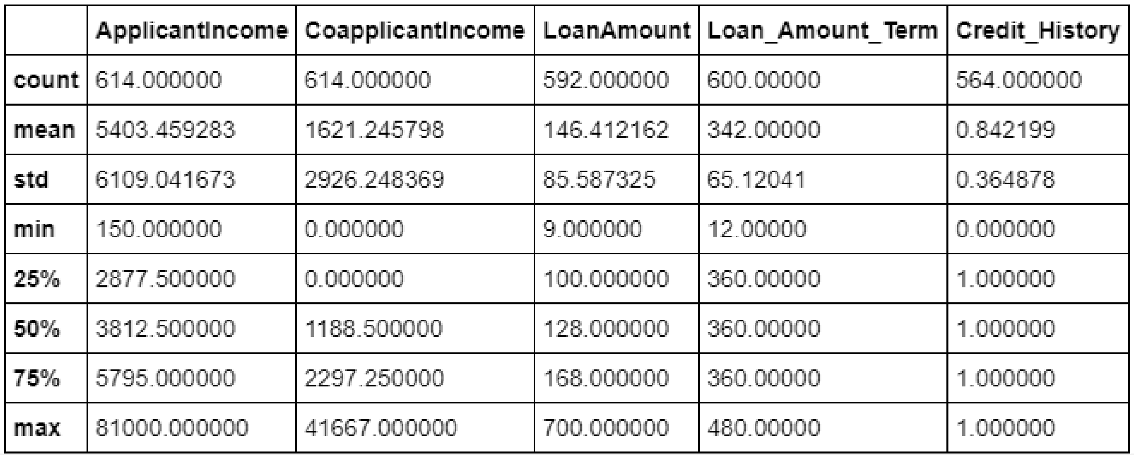
```
df_length =len(df)
# Store total number of columns in testing data set
test_col = len(test.columns)
```

7.1 Understanding the various features (columns) of the dataset.

In [4]: # Summary of numerical variables for training data set

```
df.describe()
```

Out[4]:



	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.00000	564.000000
mean	5403.459283	1621.245798	146.412162	342.00000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.00000	0.000000
25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

For the non-numerical values (e.g. Property_Area, Credit_History etc.), we can look at frequency distribution to understand whether they make sense or not.

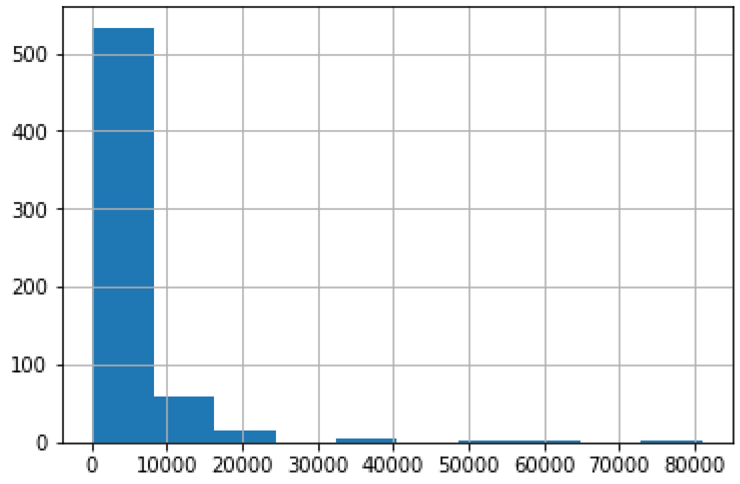
In [5]:
```
# Get the unique values and their frequency of variable Property_Area
df['Property_Area'].value_counts()
```

Out[5]:
```
Semiurban    233
Urban        202
Rural        179
Name: Property_Area, dtype: int64
```

1. Understanding Distribution of Numerical Variables
   1. ApplicantIncome
   2. LoanAmount

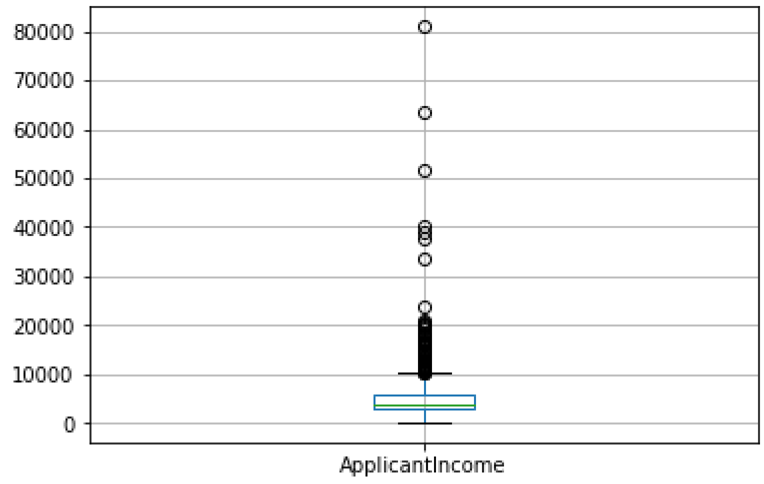
In [6]:
```
# Box Plot for understanding the distributions and to observe the outliers.
%matplotlib inline
# Histogram of variable ApplicantIncome
df['ApplicantIncome'].hist()
```

Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x11719e898>



In [7]:
```
# Box Plot for variable ApplicantIncome of training data set
df.boxplot(column='ApplicantIncome')
```

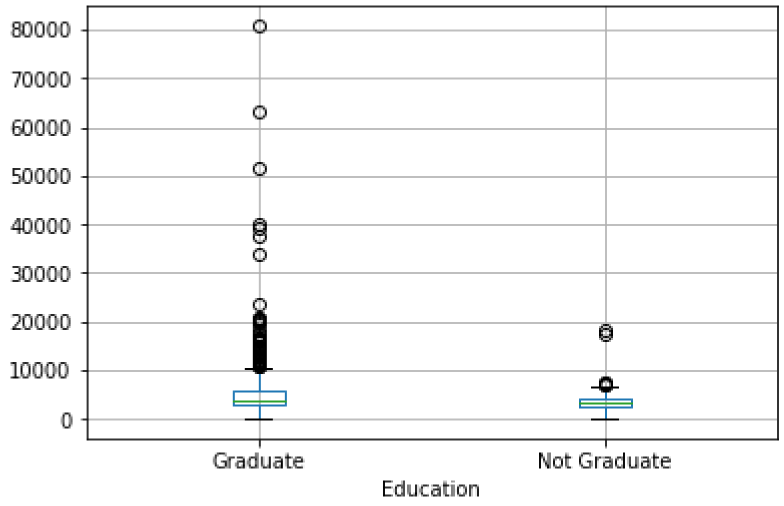
Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x1172d2da0>



The above Box Plot confirms the presence of a lot of outliers/extreme values. This can be attributed to the income disparity in the society

```
In [8]:  # Box Plot for variable ApplicantIncome by variable Education of training data set
         df.boxplot(column='ApplicantIncome', by = 'Education')
```

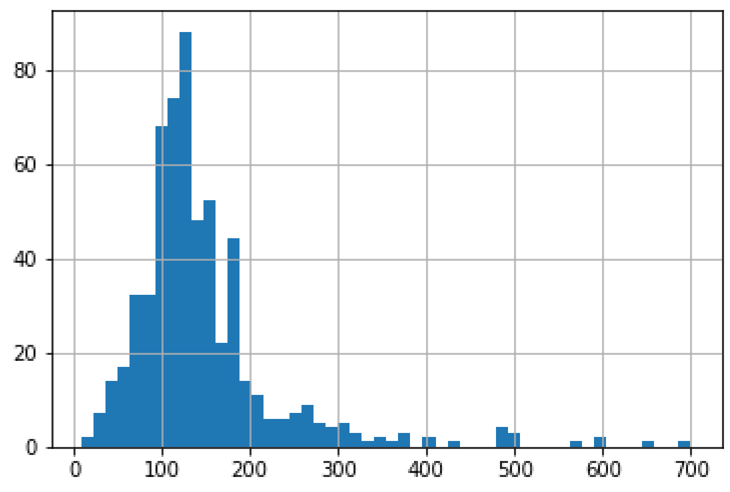
Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x117263e48>



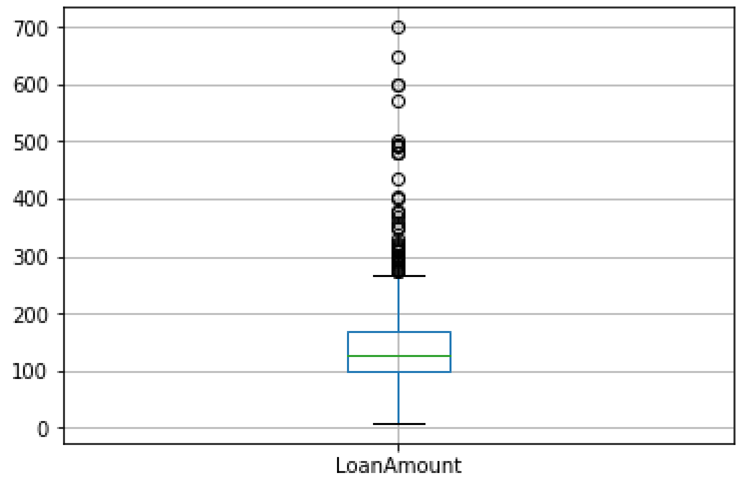
Boxplot grouped by Education ApplicationIncome

We can see that there is no substantial different between the mean income of graduate and non-graduates. But there are a higher number of graduates with very high incomes, which are appearing to be the outliers

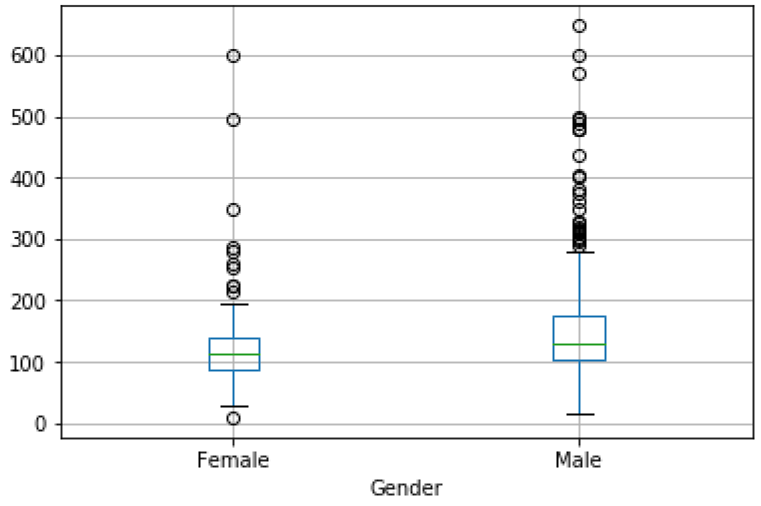
```
In [9]:  # Histogram of variable LoanAmount
         df['LoanAmount'].hist(bins=50)
Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x1173a1128>
```



```
In [10]:  # Box Plot for variable LoanAmount of training data set
          df.boxplot(column='LoanAmount')
Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x117693940>
```



```
In [11]:  # Box Plot for variable LoanAmount by variable Gender of training data set
          df.boxplot(column='LoanAmount', by = 'Gender')
Out[11]:  <matplotlib.axes._subplots.AxesSubplot at 0x11763cb38>
```



Boxplot grouped by Gender LoanAmount

LoanAmount has missing as well as extreme values, while ApplicantIncome has a few extreme values.

7.2 Understanding Distribution of Categorical Variables

In [12]: # Loan approval rates in absolute numbers

```
loan_approval = df['Loan_Status'].value_counts()['Y']
print(loan_approval)
```

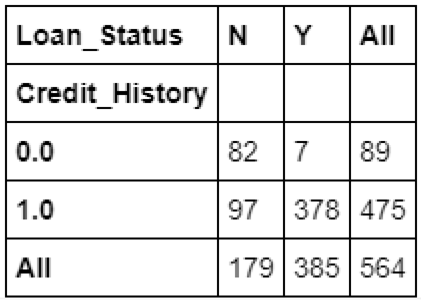
```
422
```

422 number of loans were approved.

In [13]: # Credit History and Loan Status

```
pd.crosstab(df ['Credit_History'], df ['Loan_Status'], margins=True)
```

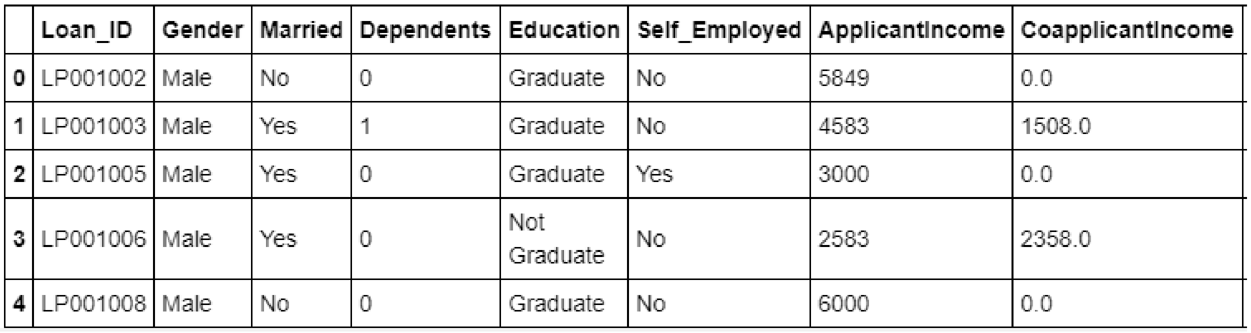
Out[13]:



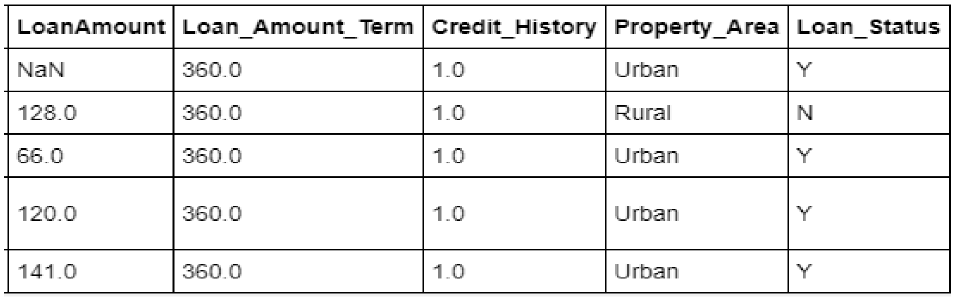
Loan_Status	N	Y	All
Credit_History			
0.0	82	7	89
1.0	97	378	475
All	179	385	564

In [16]: #Function to output percentage row wise in a cross table

```
def percentageConvert(ser):
        return ser/float(ser[-1])
        79.58 % of the applicants whose loans were approved have Credit_History equals
        to 1.
In [17]:  df.head()
Out[17]:
```



	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome
0	LP001002	Male	No	0	Graduate	No	5849	0.0
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0
4	LP001008	Male	No	0	Graduate	No	6000	0.0



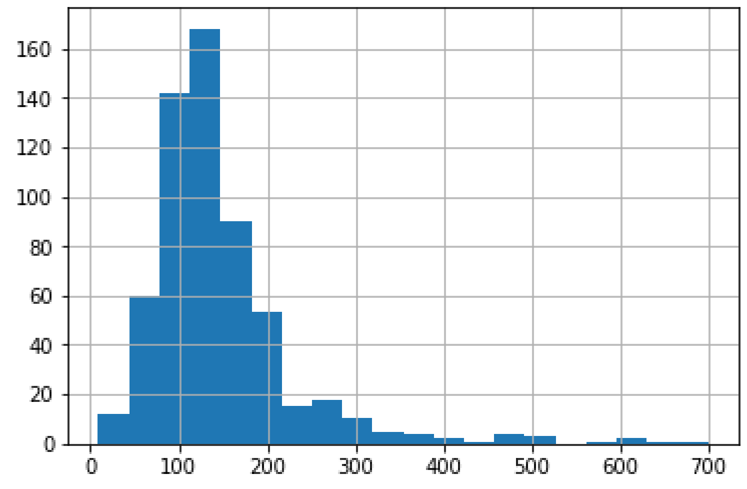
LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
NaN	360.0	1.0	Urban	Y
128.0	360.0	1.0	Rural	N
66.0	360.0	1.0	Urban	Y
120.0	360.0	1.0	Urban	Y
141.0	360.0	1.0	Urban	Y

```
In [18]:  # Replace missing value of Self_Employed with more frequent
           category
           df['Self_Employed'].fillna('No',inplace=True)
```

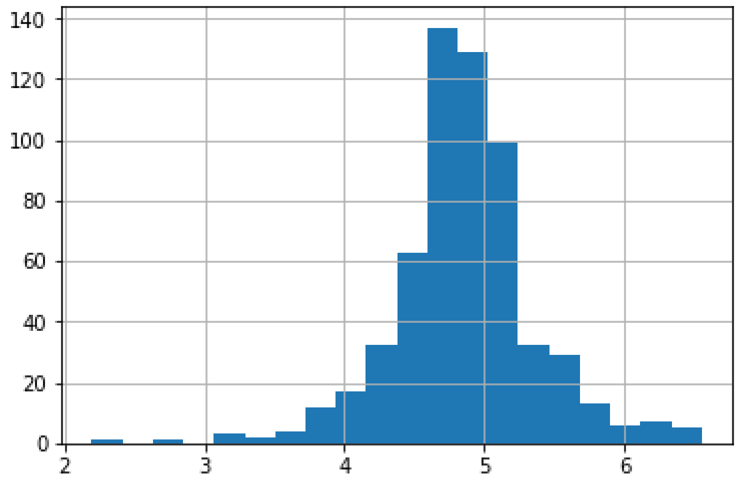
7.3 Outliers of LoanAmount and Applicant Income

```
In [19]:  # Add both ApplicantIncome and CoapplicantIncome to TotalIncome
          df['TotalIncome'] = df['ApplicantIncome'] + df['CoapplicantIncome']
          # Looking at the distribtion of TotalIncome
          df['LoanAmount'].hist(bins=20)
```

```
Out[19]:  <matplotlib.axes._subplots.AxesSubplot at 0x117cd6e48>
```



```
In [20]: # Perform log transformation of TotalIncome to make it closer to normal
         df['LoanAmount_log'] = np.log(df['LoanAmount'])
         # Looking at the distribtion of TotalIncome_log
         df['LoanAmount_log'].hist(bins=20)
Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x117dd07b8>
```



7.4 Data Preparation for Model Building

1. sklearn requires all inputs to be numeric, we should convert all our categorical variables into numeric by encoding the categories. Before that we will fill all the missing values in the dataset.

In [21]:
```
# Impute missing values for Gender
df['Gender'].fillna(df['Gender'].mode()[0],inplace=True)
# Impute missing values for Married
df['Married'].fillna(df['Married'].mode()[0],inplace=True)
# Impute missing values for Dependents
df['Dependents'].fillna(df['Dependents'].mode()[0],inplace=True)
# Impute missing values for Credit_History
df['Credit_History'].fillna(df['Credit_History'].mode()[0],inplace=True)
# Convert all non-numeric values to number
cat=['Gender','Married','Dependents','Education','Self_Employed','Credit_History',
'Property_Area']
for var in cat:
   le = preprocessing.LabelEncoder()
   df[var]=le.fit_transform(df[var].astype('str'))
df.dtypes
```

Out[21]:
```
Loan_ID               object
Gender                 int64
Married                int64
Dependents             int64
Education              int64
Self_Employed          int64
ApplicantIncome        int64
CoapplicantIncome    float64
LoanAmount           float64
Loan_Amount_Term     float64
Credit_History         int64
Property_Area          int64
Loan_Status           object
TotalIncome          float64
LoanAmount_log       float64
```

```
        dtype: object
        Generic Classification Function
In [22]: #Import models from scikit learn module:
         from sklearn import metrics
         from sklearn.cross_validation import KFold
         #Generic function for making a classification model and accessing performance:
        def classification_model(model, data, predictors, outcome):
           #Fit the model:
           model.fit(data[predictors],data[outcome])
           #Make predictions on training set:
           predictions = model.predict(data[predictors])
           #Print accuracy
           accuracy = metrics.accuracy_score(predictions,data[outcome])
           print ("Accuracy : %s" % "{0:.3%}".format(accuracy))
           #Perform k-fold cross-validation with 5 folds
           kf = KFold(data.shape[0], n_folds=5)
           error = []
           for train, test in kf:
               # Filter training data
               train_predictors = (data[predictors].iloc[train,:])
               # The target we're using to train the algorithm.
               train_target = data[outcome].iloc[train]
               # Training the algorithm using the predictors and target.
               model.fit(train_predictors, train_target)
               #Record error from each cross-validation run
               error.append(model.score(data[predictors].iloc[test,:], data[outcome].iloc[test]))
          print ("Cross-Validation Score : %s" % "{0:.3%}".format(np.mean(error)))
          #Fit the model again so that it can be refered outside the function:
          model.fit(data[predictors],data[outcome])
```

7.5 Model Building

In [23]:
```
#Combining both train and test dataset
#Create a flag for Train and Test Data set
df['Type']='Train'
test['Type']='Test'
fullData = pd.concat([df,test], axis=0)
#Look at the available missing values in the dataset
fullData.isnull().sum()
```

Out[23]:
```
ApplicantIncome          0
CoapplicantIncome        0
Credit_History          29
Dependents              10
Education                0
Gender                  11
LoanAmount              27
LoanAmount_log         389
Loan_Amount_Term        20
Loan_ID                  0
Loan_Status            367
Married                  0
Property_Area            0
Self_Employed           23
TotalIncome            367
Type                     0
dtype: int64
```

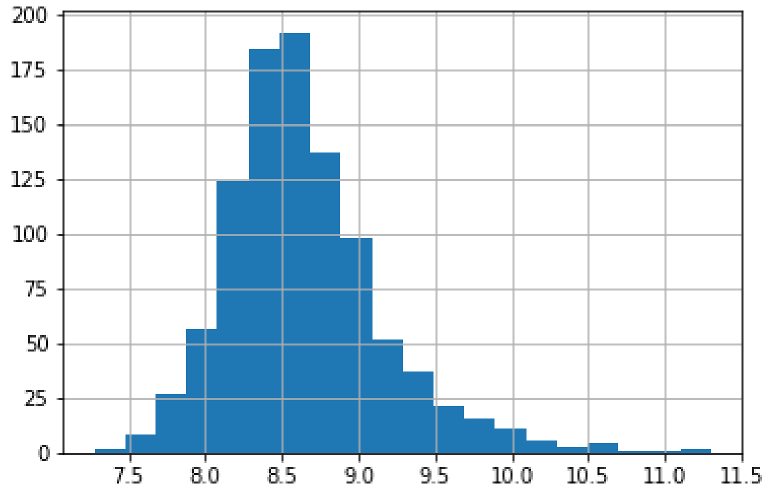
In [24]:
```
#Identify categorical and continuous variables
ID_col = ['Loan_ID']
target_col = ["Loan_Status"]
```

```
cat_cols = ['Credit_History','Dependents','Gender','Married','Education','Property_Area',
'Self_Employed']
```

In [25]:
```
#Imputing Missing values with mean for continuous variable
fullData['LoanAmount'].fillna(fullData['LoanAmount'].mean(), inplace=True)
fullData['LoanAmount_log'].fillna(fullData['LoanAmount_log'].mean(), inplace=True)
fullData['Loan_Amount_Term'].fillna(fullData['Loan_Amount_Term'].mean(), inplace=
True)
fullData['ApplicantIncome'].fillna(fullData['ApplicantIncome'].mean(), inplace=True)
fullData['CoapplicantIncome'].fillna(fullData['CoapplicantIncome'].mean(), inplace=True
)
#Imputing Missing values with mode for categorical variables
fullData['Gender'].fillna(fullData['Gender'].mode()[0], inplace=True)
fullData['Married'].fillna(fullData['Married'].mode()[0], inplace=True)
fullData['Dependents'].fillna(fullData['Dependents'].mode()[0], inplace=True)
fullData['Loan_Amount_Term'].fillna(fullData['Loan_Amount_Term'].mode()[0], inplace
=True)
fullData['Credit_History'].fillna(fullData['Credit_History'].mode()[0], inplace=True)
```

In [26]:
```
#Create a new column as Total Income
fullData['TotalIncome']=fullData['ApplicantIncome'] + fullData['CoapplicantIncome']
fullData['TotalIncome_log'] = np.log(fullData['TotalIncome'])
#Histogram for Total Income
fullData['TotalIncome_log'].hist(bins=20)
```

Out[26]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x117f1cd30>
```



```
In [27]: #create label encoders for categorical features
         for var in cat_cols:
             number = LabelEncoder()
             fullData[var] = number.fit_transform(fullData[var].astype('str'))

         train_modified=fullData[fullData['Type']=='Train']
         test_modified=fullData[fullData['Type']=='Test']
         train_modified["Loan_Status"] = number.fit_transform(train_modified["Loan_Status"].as
         type('str'))
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
/Users/Technocrat/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:8: SettingWith
CopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html
#indexing-view-versus-copy
```

7.6 Logistic Regression Model

The chances of getting a loan will be higher for:

1. Applicants having a credit history (we observed this in exploration.)
2. Applicants with higher applicant and co-applicant incomes
3. Applicants with higher education level
4. Properties in urban areas with high growth perspectives

So let's make our model with 'Credit_History', 'Education' & 'Gender'

```
In [28]: from sklearn.linear_model import LogisticRegression
         predictors_Logistic=['Credit_History','Education','Gender']
         x_train = train_modified[list(predictors_Logistic)].values
        y_train = train_modified["Loan_Status"].values
        x_test=test_modified[list(predictors_Logistic)].values
```

```
In [29]: # Create logistic regression object
         model = LogisticRegression()
         # Train the model using the training sets
         model.fit(x_train, y_train)
         #Predict Output
         predicted= model.predict(x_test)
         #Reverse encoding for predicted outcome
         predicted = number.inverse_transform(predicted)
         #Store it to test dataset
         test_modified['Loan_Status']=predicted
         outcome_var = 'Loan_Status'
         classification_model(model, df,predictors_Logistic,outcome_var)
         test_modified.to_csv("Logistic_Prediction.csv",columns=['Loan_ID','Loan_Status'])
```

```
         Accuracy : 80.945%

         Cross-Validation Score : 80.946%
/Users/Technocrat/anaconda3/lib/python3.6/site-packages/sklearn/preprocessing/label.py:151:
DeprecationWarning: The truth value of an empty array is ambiguous. Returning False, but in
```

```
future this will result in an error. Use `array.size > 0` to check that an array is not empty.
if diff:
/Users/Technocrat/anaconda3/lib/python3.6/site-packages/ipykernel_launcher.py:14:SettingWith
CopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
```

CHAPTER-7

CONCLUSION

After trying and testing 4 different algorithms, the best accuracy on the public leaderboard is achieved by Logistic Regression (0.7847), followed by RandomForest (0.7778) and XGBoost (0.7778), and Decision Tree performed the worst (0.6458). While new features created via feature engineering helped in predicting the target variable, it did not improve the overall model accuracy much. Compared to using default parameter values, GridSearchCV helped improved the model's mean validation accuracy by providing the optimized values for the model's hyperparameters. On the whole, a logistic regression classifier provides the best result in terms of accuracy for the given dataset, without any feature engineering needed. Because of its simplicity and the fact that it can be implemented relatively easy and quick, Logistic Regression is often a good baseline that data scientists can use to measure the performance of other more complex algorithms. In this case, however, a basic Logistic Regression has already outperformed other more complex algorithms like Random Forest and XGBoost, for the given dataset.

Suggestions for Improvement. There are many things that can be tried to improve the models' predictions. We can create and add more variables, try different models with different subset of features and/or rows, etc. Some of the ideas are listed below:

1. Combine the applicants with 1,2,3 or more dependents and make a new feature as discussed in the EDA part.
2. Make independent vs independent variable visualizations to discover some more patterns.
3. Arrive at the EMI using a better formula which may include interest rates as well.
4. Try ensemble modeling (combination of different models). More about ensemble techniques can be found at the references.

Try neural network using Tensorflow or PyTorch

CHAPTER-8

REFERENCES

1. https://towardsdatascience.com/data-types-in-statistics-347e152e8bee
2. https://machinelearning-blog.com/2018/04/23/logistic-regression-101/
3. https://www.analyticsvidhya.com/blog/2015/08/introduction-ensemble-learning/
4. https://www.analyticsvidhya.com/blog/2015/09/questions-ensemble-modeling/