SUBJECT- INT 247(Term Paper)

Topic – Lone Prediction analysis

Name – Subham Nandi

Subject- Machine Learning Foundation

Roll No- 24

Section- K19GT

Acknowledgement

I would like to thanks our mentor Dr Sagar Pande for his advice and inputs on this projects.Under her guidance we made the project .Many many thanks to Dr Sagar Pande.

Motivation

Being extremely interested in everything having a relation with the Machine Learning, the group project was a great occasion to give us the time to learn and confirm our interest for this field. The fact that we can make estimations, predictions and give the ability for machines to learn by themselves is both powerful and limitless in term of application possibilities. We can use Machine Learning in Finance, Medicine, almost everywhere. That’s why I decided to conduct my project around the Machine Learning.

Abstract

With the enhancement in the banking sector lots of people are applying for bank loans but the bank has its limited assets which it has to grant to limited people only, so finding out to whom the loan can be granted which will be a safer option for the bank is a typical process. So in this paper we try to reduce this risk factor behind selecting the safe person so as to save lots of bank efforts and assets. This is done by mining the Big Data of the previous records of the people to whom the loan was granted before and on the basis of these records/experiences the machine was trained using the machine learning model which give the most accurate result. The main objective of this paper is to predict whether assigning the loan to particular person will be safe or not. This paper is divided into four sections (i)Data Collection (ii) Comparison of machine learning models on collected data (iii) Training of system on most promising model (iv) Testing.

Problem Statement:

Finance companies, banks are deals with different kinds of loans such as education loan, shop loans, home loans, personal loans etc all are part of our country loan types. All the companies and banks are present in villages, towns, cities. After customer apply for loan these banks/companies want to validate the customer details for that candidate eligible for loan or not. The main purpose of the system is applicant loan approved or not based on train models.

Introduction

Circulation of the loans is that the core business a part of as resources are straightforwardly came from the benefit acquire from the advances distributed by the banks. The main goal in banking system is to invest their resources in safe hands wherever after a relapse right candidate out of all candidates. Through this method we are able to predict whether that particular candidate is safe or not and the whole method of validation of attribute is automated by machine learning technique. The disadvantage of this model is that it emphasizes completely different weights to every issue however, sometime loan can be approved on the premise Loan Prediction is useful for member of staff of banks as well as for the candidate. The aim of this Paper is to apply quick, immediate and easy way to choose the worthy person. It will give special gain to the bank. The Loan Prediction method can automatically compute the heaviness of each attribute taking part in loan processing and on new test data information same issues . are prepared with regard to their comparable heaviness. A period breaking point can be set for the candidate to check regardless of whether his/her loan can be affirmed or not. Loan Prediction technique licenses bouncing to explicit candidates with the goal that it very well may be keep an eye on need premise. This Paper is completely overseeing the power of Bank/finance Company, entire procedure of prediction is done secretly no colleagues would have the option to caution the process. Result against specific Loan Id can be ship off different divisions of companies so that they can make a proper move on application. This aides all others divisions to done different conventions. Data Source we obtained customer loan dataset from Kaggle . The dataset consists of various values/variables such as sex, marital status, education, self employed, loan status, applicant income, etc…Data Description the dataset has 614 rows and 13 columns. We will then decide whether the applicant is suitable for the loan based on the interest rate.

Lenders (investors) make loans to creditors in return for the guarantee of interest-bearing repayment. That is, the lender only makes a return (interest) if the borrower repays the loan. However, whether he or she does not repay the loan, the lender loses money. Banks make loans to customers in exchange for the guarantee of repayment. Some would default on their debts, unable to repay them for a number of reasons. The bank retains insurance to minimize the possibility of failure in the case of a default. The insured sum can cover the whole loan amount or just a portion of it. Banking processes use manual procedures to determine whether or not a borrower is suitable for a loan based on results. Manual procedures were mostly effective, but they were insufficient when there were a large number of loan applications. At that time, making a decision would take a long time. As a result, the loan prediction machine learning model can be used to assess a customer's loan status and build strategies. This model extracts and introduces the essential features of a borrower that influence the customer's loan status. Finally, it produces the planned performance (loan status). These reports make a bank manager's job simpler and quicker.

Literature Review :

We start our literature review with more general systematic literature reviews that focus on the application of machine learning in the general field of Banking Risk Management. Since the global financial crisis, risk management in banks has to take a major role in shaping decision-making for banks. A major portion of risk management is the approval of loans to promising candidates. But the black-box nature of Machine learning algorithms makes many loan providers vary the result. Martin Leo, Suneel Sharma and k. Maddulety's extensive report has explored where Machine Learning is being used in the fields of credit risk, market risk, operational risk, and liquidity risk only to conclude that the research falls short of extensive research is required in the field. We could not find any literature review for loan

prediction for specific Machine learning algorithms to use which would be a possible starting point for our paper. Instead, since loan prediction is a classification problem, we went with popular classification algorithms used for a similar problem. Ashlesha Vaidya [2] used logistic regression as a probabilistic and predictive approach to loan approval prediction. The author pointed out how Artificial neural networks and Logistic regression are most used for loan prediction as they are easier comparatively develop and provide the most accurate predictive analysis. One of the reasoning behind this that that other Algorithms are generally bad at predicting from non-normalized data. But the nonlinear effect and power terms are easily handled by Logistic regression as there is no need for the independent variables on which the prediction takes place to be normally distributed.

Packages used :

NumPy:- NumPy is a general-purpose array-processing package. It provides a high performance multidimensional array object, and tools for working with these arrays.It is the fundamental package for scientific computing with Python. As the whole project is based on whole complex stats ,we will use this fast calculations and provide results.

Pandas: - Pandas is the most popular python library that is used for data analysis. We will provide highly optimized performance with back-end source code with the use of Pandas.

Matplotlib:- Matplotlib tries to make easy things easy and hard things possible. We will generate plots, histograms, scatterplots, etc.,to make our project more appealing and easier to understand.

Seaborn:- We will use it for statistical data visualization as Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

Scikit-learn:- It is a Python library is associated with NumPy and SciPy. It is considered as one of the best libraries for working with complex data. There are a lot of changes being made in this library. We will use it for cross validation feature, providing the ability to use more than one metric. Lots of training methods like logistics regression will be used to provide some little improvements.

About The Dataset:

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. This is a standard supervised classification task.A classification problem where we have to predict whether a loan would be approved or not. Below is the dataset attributes with description.

Variable	Description
Loan_ID	Unique Loan ID
Gender	Male/ Female
Married	Applicant married (Y/N)
Dependents	Number of dependents
Education	Applicant Education (Graduate/ Under Graduate)
Self-employed	Self employed (Y/N)
Applicant Income	Applicant income

Variable	Description
CoapplicantIncome	Coapplicant income
Loan Amount	Loan amount in thousands
Loan_Amount_Term	Term of loan in months
Credit History	credit history meets guidelines
Property Area	Urban/ Semi Urban/ Rural
Loan Status	Loan approved (Y/N)

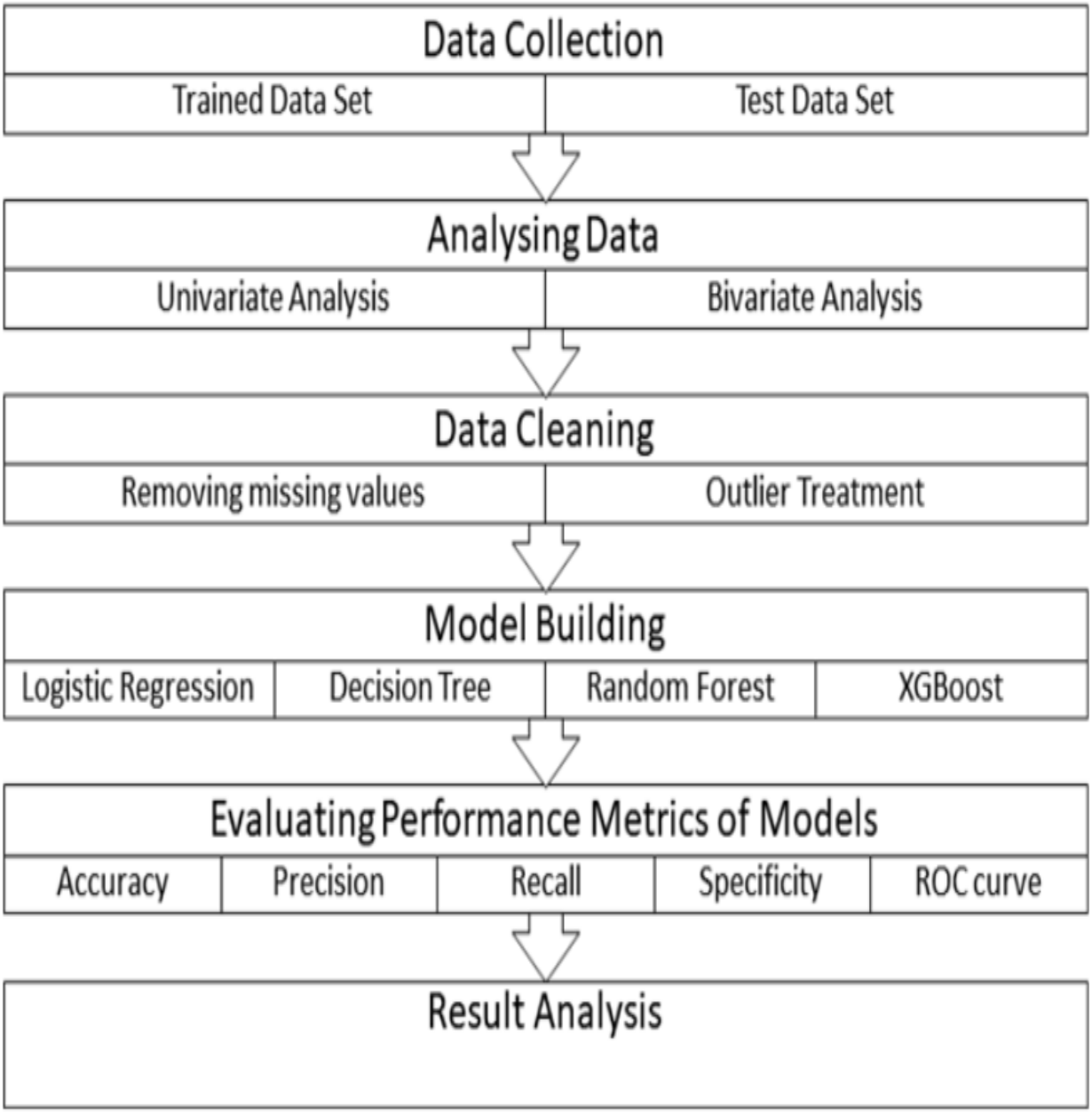
Proposed model:

In Machine Learning, we are using semi-automated extraction of knowledge of data for identifying whether a loan would be approved or not. Classification could be a supervised learning within which the response is categorical that's its values area unit in finite unordered set. To easily the matter of classification, scikit learn are used. The praim primacy of this system is company need not has to maintain a ground team to validate and verify the customer records. They can easily check whether the loan has to be approved or not by this prediction model.

In this paper we try to develop user interface flexibly graphics concepts in mind, associated through a browser interface. Our goal is to implement machine learning model so as to classify, to the best potential degree of accuracy, master card fraud from a dataset gathered from Kaggle. once initial knowledge exploration, we have a tendency to knew we might implement a random forest model for best accuracy reports. Random forest, as it was a good candidate for binary classification. Python sklearn library was used to implement the project, We used Kaggle datasets for Credit card fraud detection, using pandas to data frame for class ==0 for no fraud and class==1 for fraud, matplotlib for plotting the fraud and non fraud data, train_test_split for data extraction (Split arrays or matrices into random train and test subsets) and used Logistic Regression

machine learning algorithm for fraud detection and print predicting score according to the algorithm. Finally Confusion matrix was plotted on true and predicted.

Methodology:



Implementation Details:

Loan Dataset : Loan Dataset is very useful in our system for prediction of more accurate result. Using the loan Dataset the system will automatically predict which costumer's loan it should approve and which to reject. System will accept loan application form as an input. Justified format of application form should be given as an input to get processed.

Determine the training and testing data: Typically , Here the system separate a dataset into a training set and testing set ,most of the data use for training ,and a smaller portions of data is use for testing. after a system has been processed by using the training set, it makes the prediction against the test set.

Data cleaning and processing: InData cleaning the system detect and correct corrupt or inaccurate records from database and refers to identifying incomplete, incorrect, inaccurate or irrelevant parts of the data and then replacing , modifying or detecting the dirty or coarse data. In Data processing the system convert data from a given form to a much more usable and desired form i.e. make it more meaningful and informative.11

Dataset Information

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.

This is a standard supervised classification task.A classification problem where we have to predict whether a loan would be approved or not. Below is the dataset attributes with description.

Variable	Description
Loan_ID	Unique Loan ID
Gender	Male/ Female
Married	Applicant married (Y/N)
Dependents	Number of dependents
Education	Applicant Education (Graduate/ Under Graduate)
Self_Employed	Self employed (Y/N)
ApplicantIncome	Applicant income
CoapplicantIncome	Coapplicant income
LoanAmount	Loan amount in thousands
Loan_Amount_Term	Term of loan in months
Credit_History	credit history meets guidelines
Property_Area	Urban/ Semi Urban/ Rural
Loan_Status	Loan approved (Y/N)

Import modules

```
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
import matplotlib
%matplotlib inline
```

Loading the Dataset

```
df = pd.read_csv("Loan Prediction Dataset.csv")
df.head()
```

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

- We have to predict the output variable "Loan status".
- The Input attributes are in categorical as well as in numerical form.
- We have to analyze all the attributes.

Statistics Data Information

```
df.describe()
```

	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

- The total count column displays some missing values, which we will deal with later.
- The credit history attributes are in the range of 0 to 1.

df.info()

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

- We can observe 13 attributes. Out of which 4 attributes are in float, 1 attribute is in integer and the other 8 are in objects.
- We can change the object into corresponding data to reduce the usage memory.
- However, we have 62 KB of memory usage, therefore we don't have to change any of the data types.

Preprocessing the Loan Sanction Data

Let us check for NULL values in the dataset.

```
# find the null values
df.isnull().sum()
```

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

- We have found 6 columns having NULL values.
- Now, we have to replace the NULL values with some common values.

Let us fill in the missing values for numerical terms using mean.

```
# fill the missing values for numerical terms - mean df['LoanAmount'] =
df['LoanAmount'].fillna(df['LoanAmount'].mean()) df['Loan_Amount_Term'] =
df['Loan_Amount_Term'].fillna(df['Loan_Amount_Term'].mean())
df['Credit_History'] = df['Credit_History'].fillna(df['Credit_History']
```

- All the missing values will be filled with the mean of the current column.

Let us now fill in the missing values for categorical terms using mode operation.

```
# fill the missing values for categorical terms - mode df['Gender'] =
df["Gender"].fillna(df['Gender'].mode()[0]) df['Married'] =
df["Married"].fillna(df['Married'].mode()[0]) df['Dependents'] =
df["Dependents"].fillna(df['Dependents'].mode()[0]) df['Self_Employed'] =
df["Self_Employed"].fillna(df['Self_Employed'].mode()[0]
```

- All the missing values will be filled with the most frequently occurring values.
- Modes give the result in their terms of the data frame, so we only need the values. We will specify 0th index to display the values.

Now, let's check for the NULL values again.

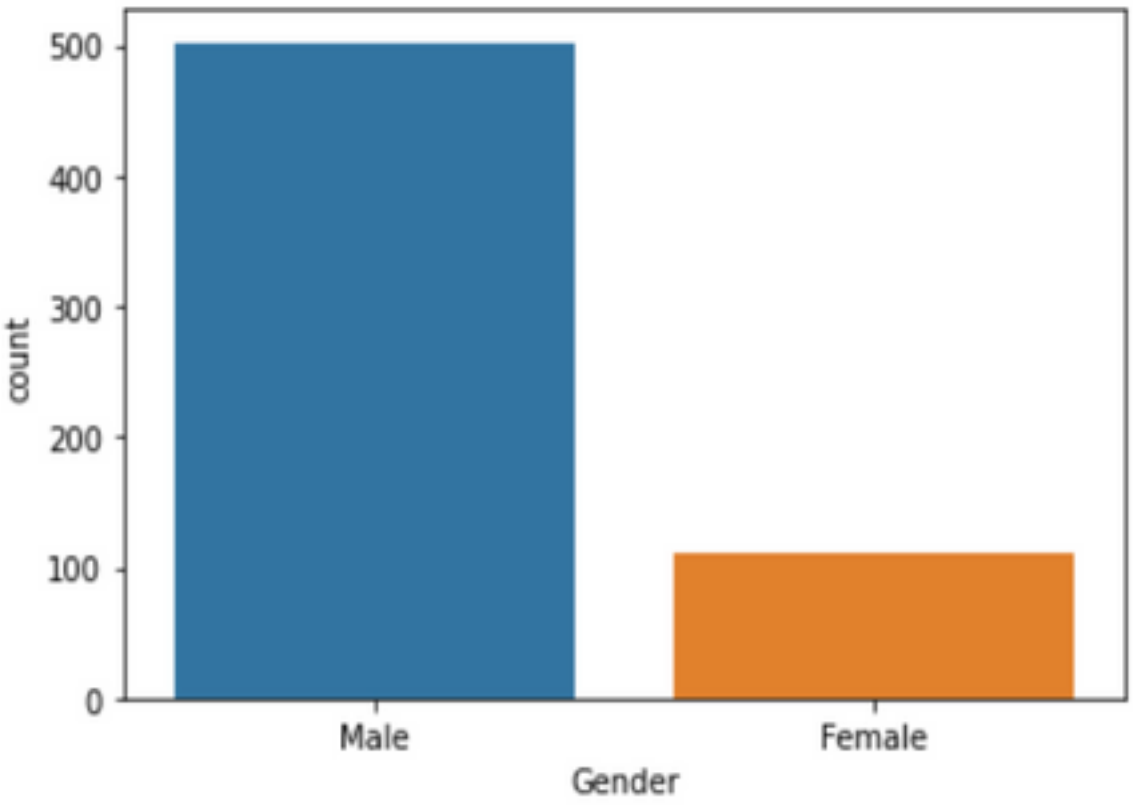
```
df.isnull().sum()
```

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

Exploratory Data Analysis

Let us first explore the categorical column "Gender".

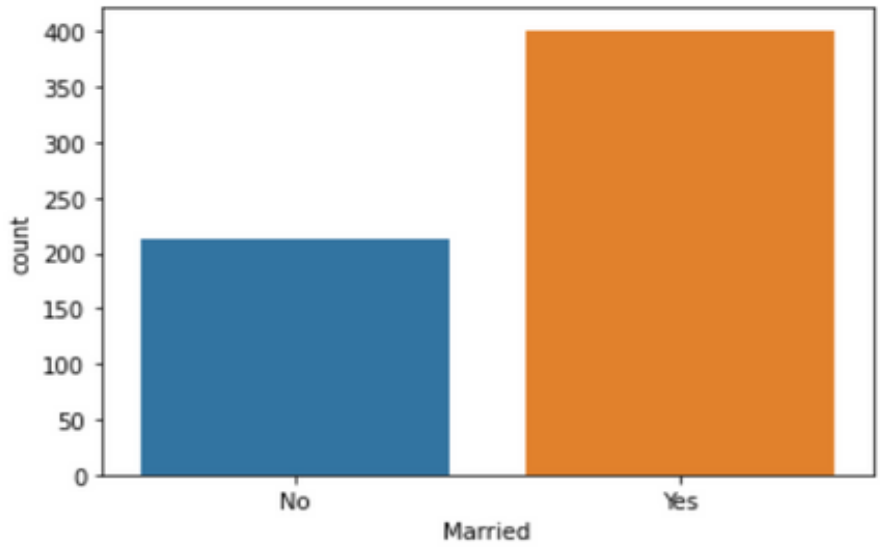
```
# categorical attributes visualization
sns.countplot(df['Gender'])
```



- The majority of the applicant is male and a handful is female.
- From these analyses, we will get an intuition that will be useful in building the model.

To display the column "Married".

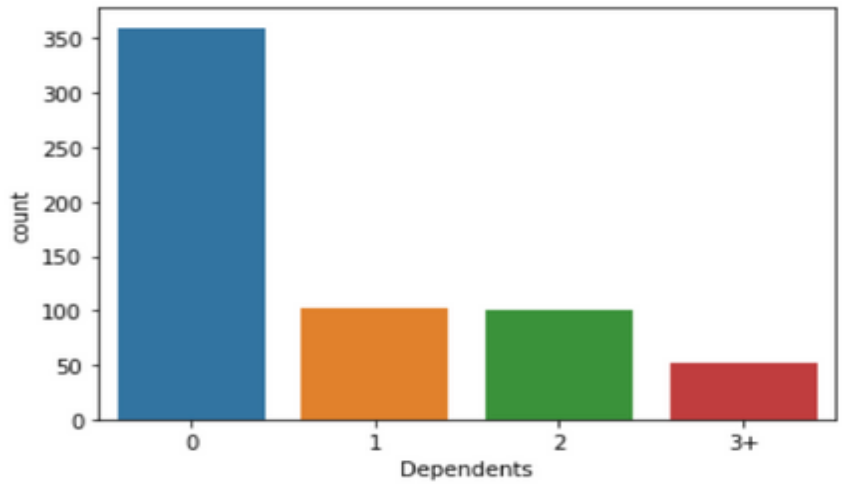
```
sns.countplot(df['Married'])
```



The majority of the applicants are married.

To display the column "Dependents".

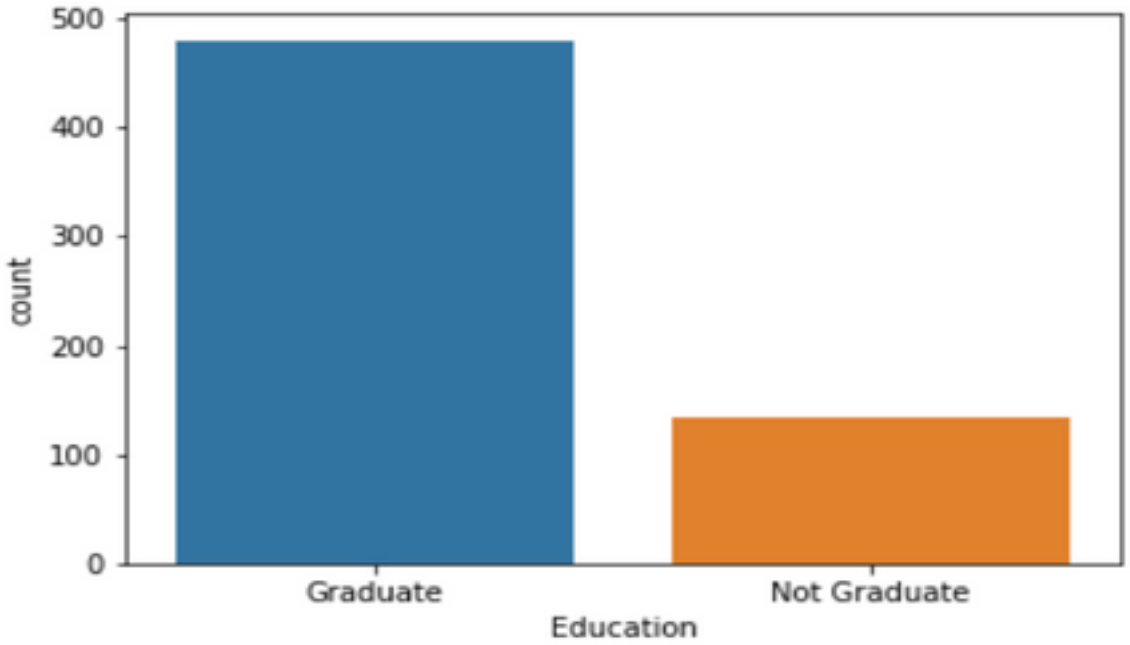
```
sns.countplot(df['Dependents'])
```



- The majority of the applicants have zero dependents, around 100 applicants have one or two dependents and only a few have more than three dependents.

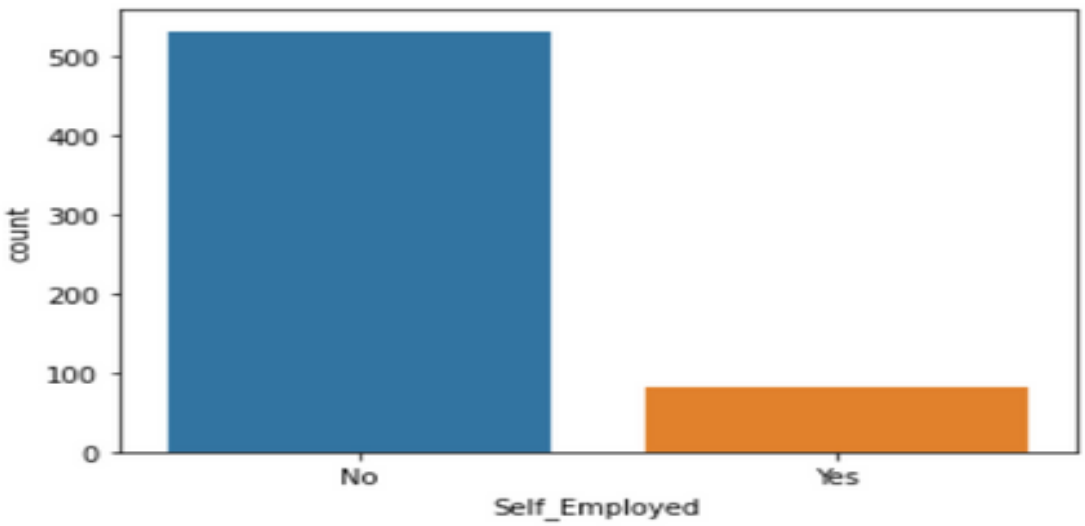
To display the column "Education".

```
sns.countplot(df['Education'])
```



To display the column "Self Employed".

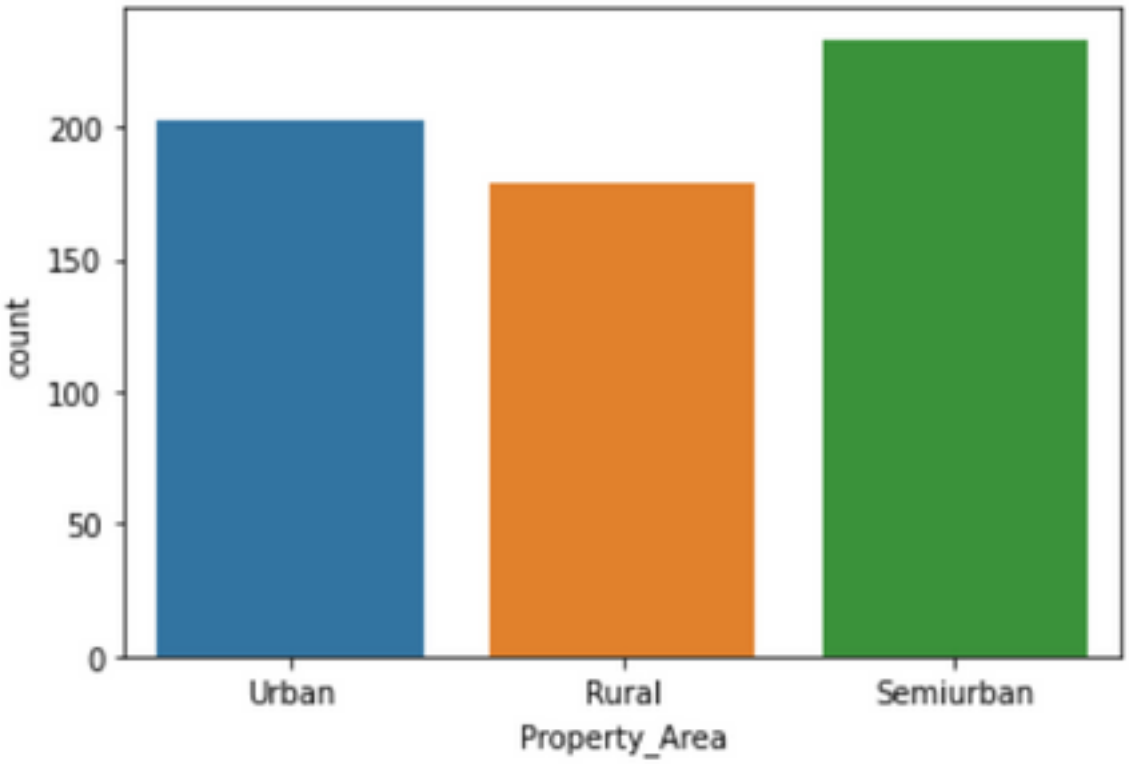
```
sns.countplot(df['Self_Employed'])
```



- Around 90 applicants are either freelancers or run a business.

To display the column "Property Area".

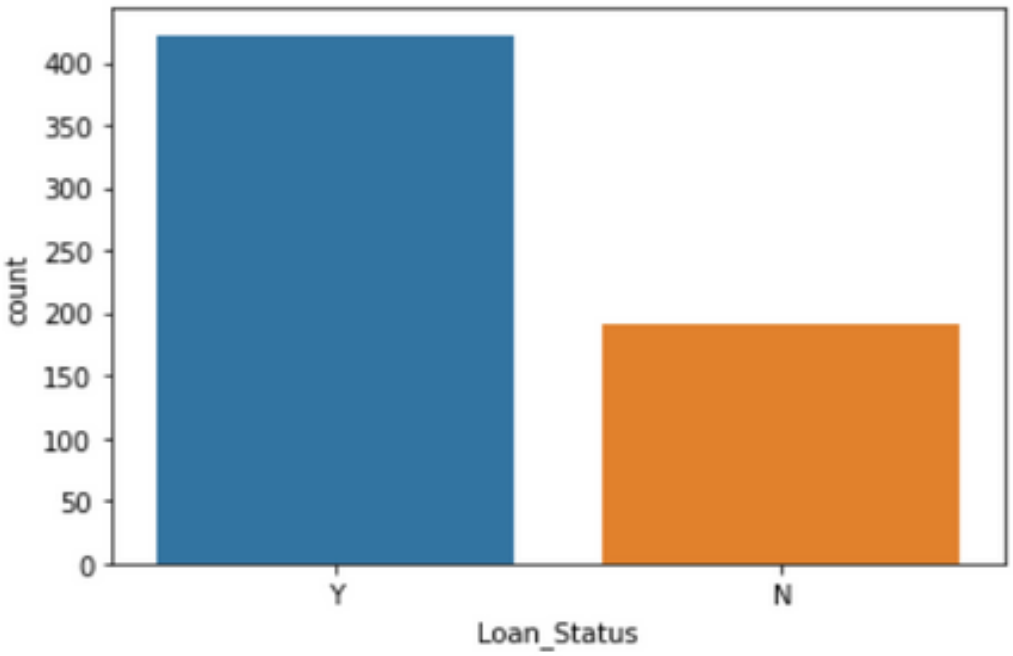
```
sns.countplot(df['Property_Area'])
```



We can assume that the applicants are equally distributed in urban, rural and semi-urban areas.

To display the column "Loan Status".

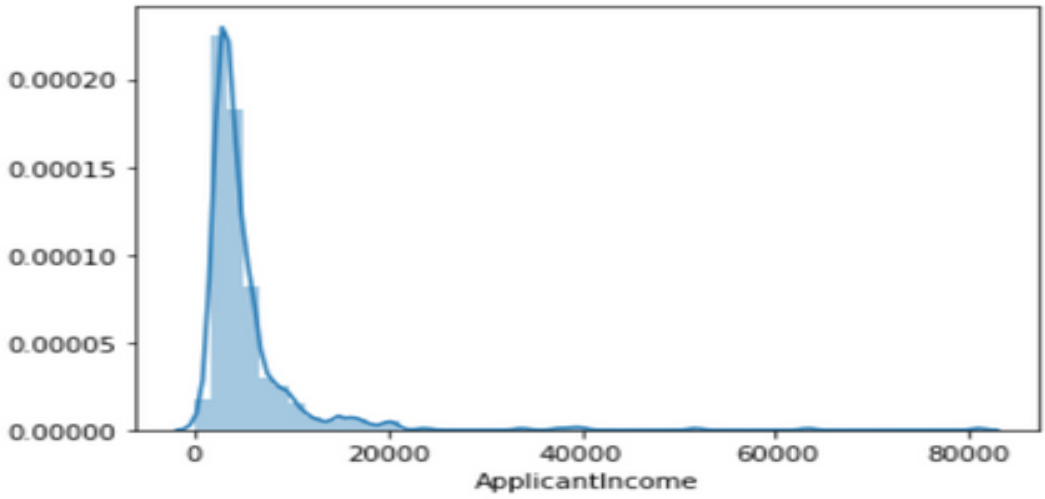
```
sns.countplot(df['Loan_Status'])
```



- Around 400 loans are accepted and 200 loans are rejected. Its shows the 2:1 ratio.

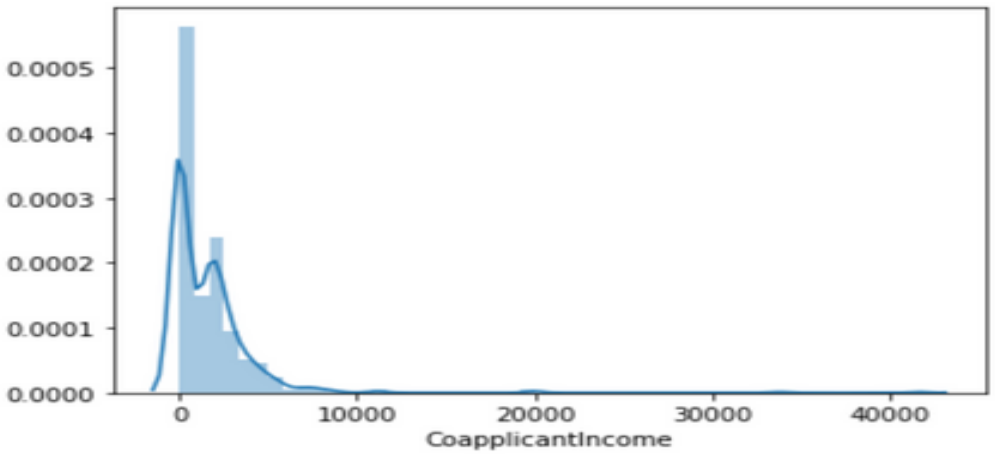
- **Let us first explore the Numerical column "Applicant Income".**
-

```
# numerical attributes visualization
sns.distplot(df["ApplicantIncome"])
```



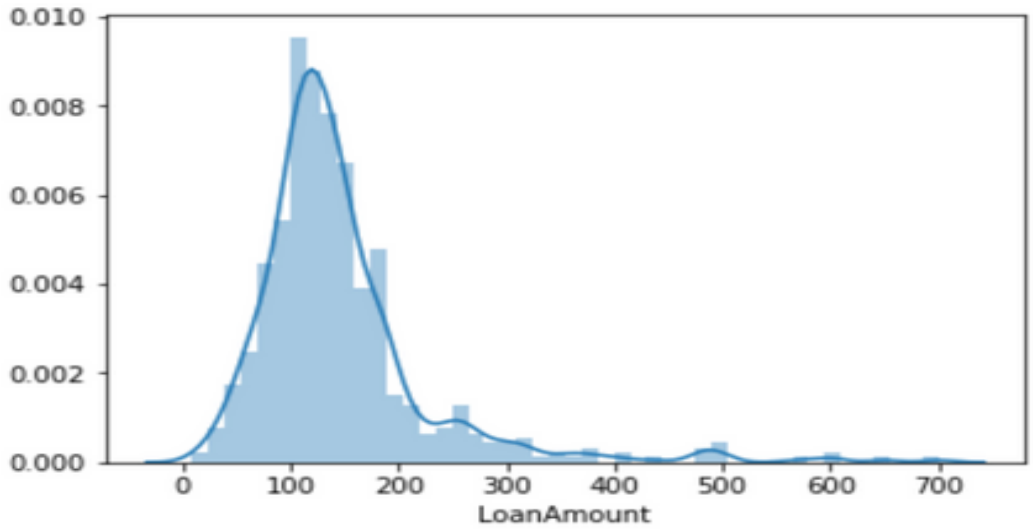
To display the column "Co-applicant Income".

```
sns.distplot(df["CoapplicantIncome"])
```



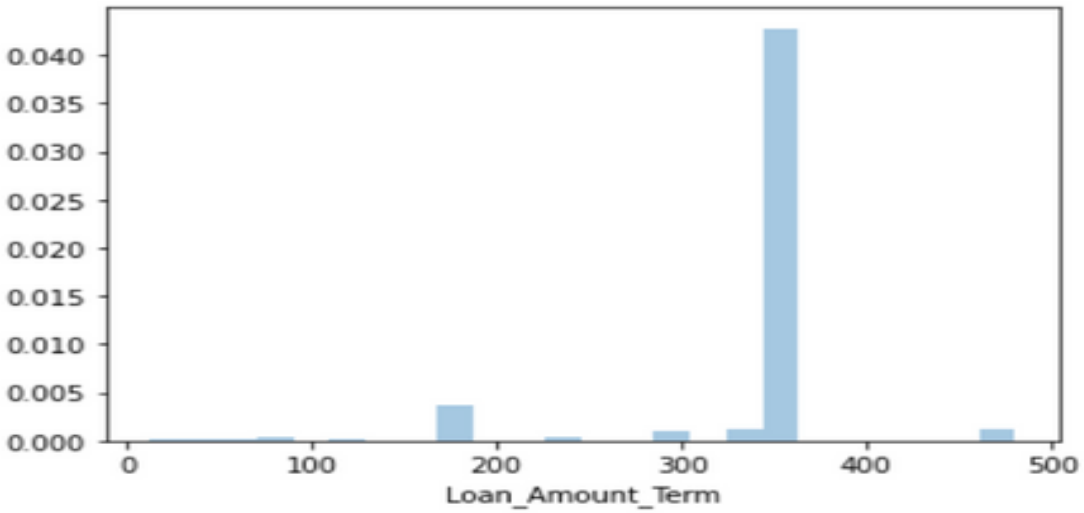
To display the column "Loan Amount".

```
sns.distplot(df["LoanAmount"])
```



display the To column "Loan Amount Term".

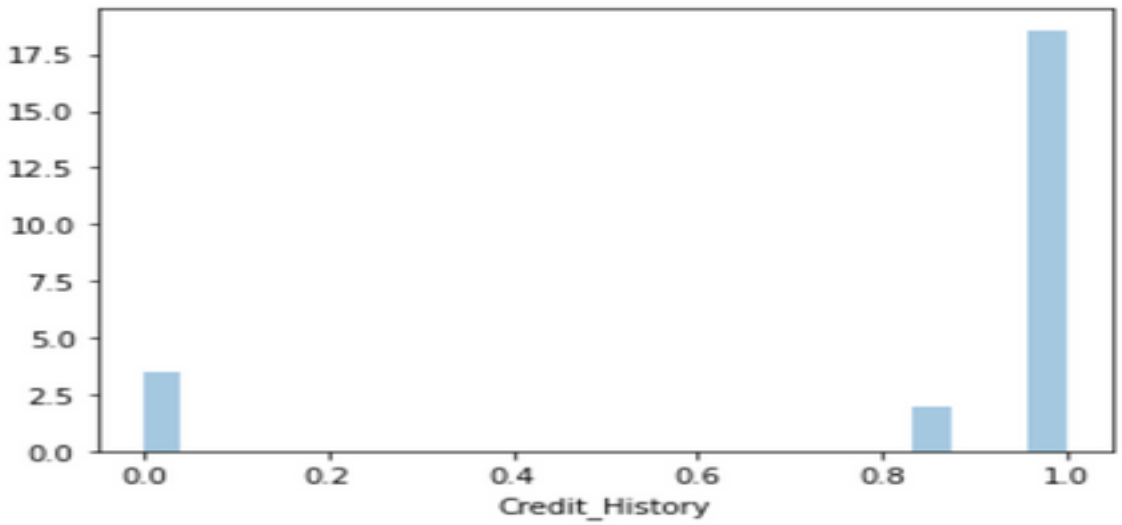
```
sns.distplot(df['Loan_Amount_Term'])
```



- The majority of them are filled will main values, that is the highest values. We will apply log transformation of this as well.

To display the column "Credit History".

```
sns.distplot(df['Credit_History'])
```



- Since the values of credit history are in the range of 0 to 1, we don't need to normalize this graph.

Creation of new attributes

We can create a new attribute performing Log Transformation. We can also create a new attribute **Total Income**, that is the sum of Applicant Income and Co-applicant Income.

```
# total income df['Total_Income'] =
df['ApplicantIncome'] + df['CoapplicantIncome']
df.head()
```

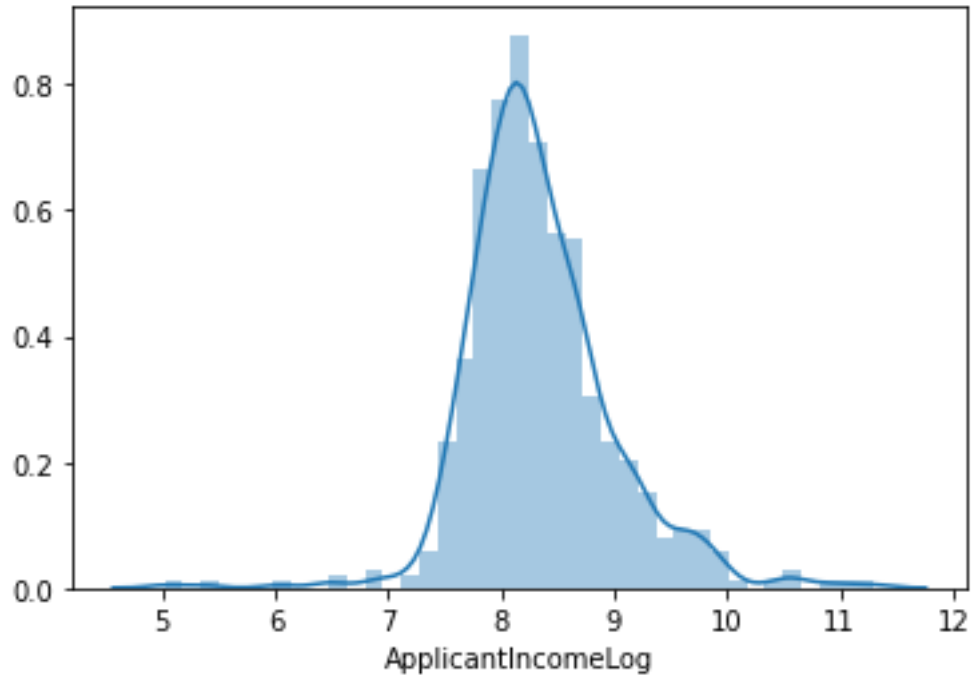
Income	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status	Total_Income
5849	0.0	146.412162	360.0	1.0	Urban	Y	5849.0
4583	1508.0	128.000000	360.0	1.0	Rural	N	6091.0
3000	0.0	66.000000	360.0	1.0	Urban	Y	3000.0
2583	2358.0	120.000000	360.0	1.0	Urban	Y	4941.0
6000	0.0	141.000000	360.0	1.0	Urban	Y	6000.0

Log Transformation

In [22]:

```
# apply log transformation to the attribute
df['ApplicantIncomeLog'] = np.log(df['ApplicantIncome']+1)
sns.distplot(df["ApplicantIncomeLog"])
```

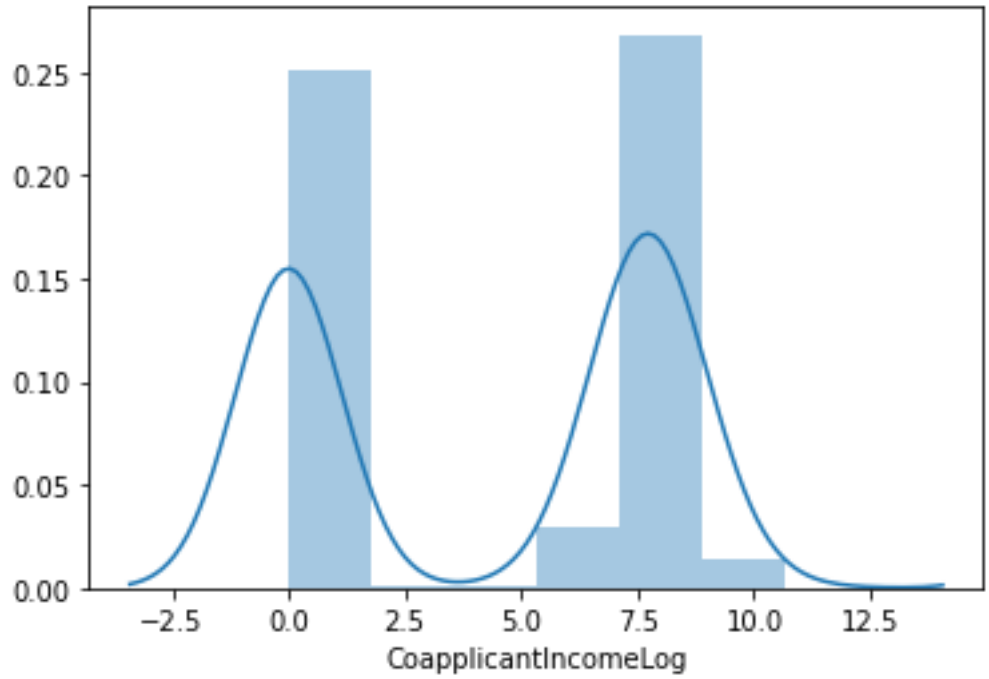
Out[22]:



In [27]:

```
df['CoapplicantIncomeLog'] =
np.log(df['CoapplicantIncome']+1)
sns.distplot(df["CoapplicantIncomeLog"])
```

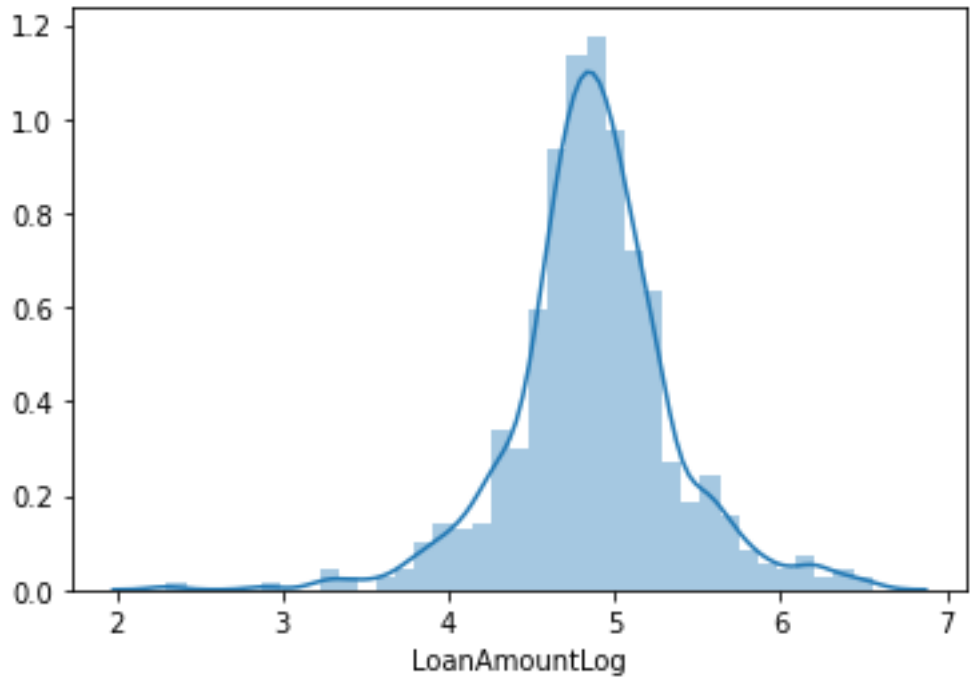
Out[27]:



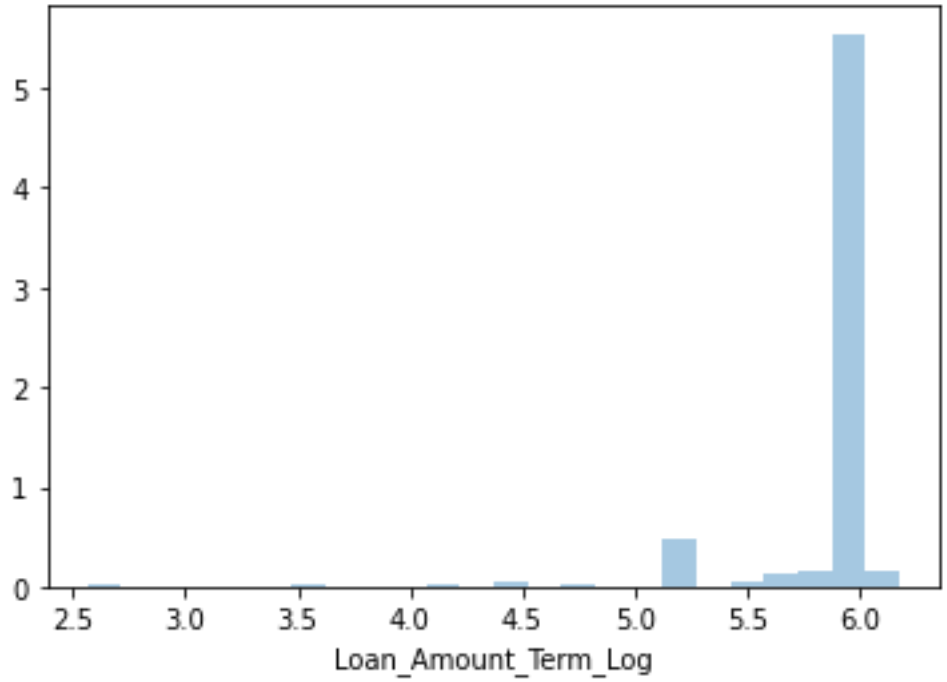
In [24]:

```
df['LoanAmountLog'] = np.log(df['LoanAmount']+1)
sns.distplot(df["LoanAmountLog"])
```

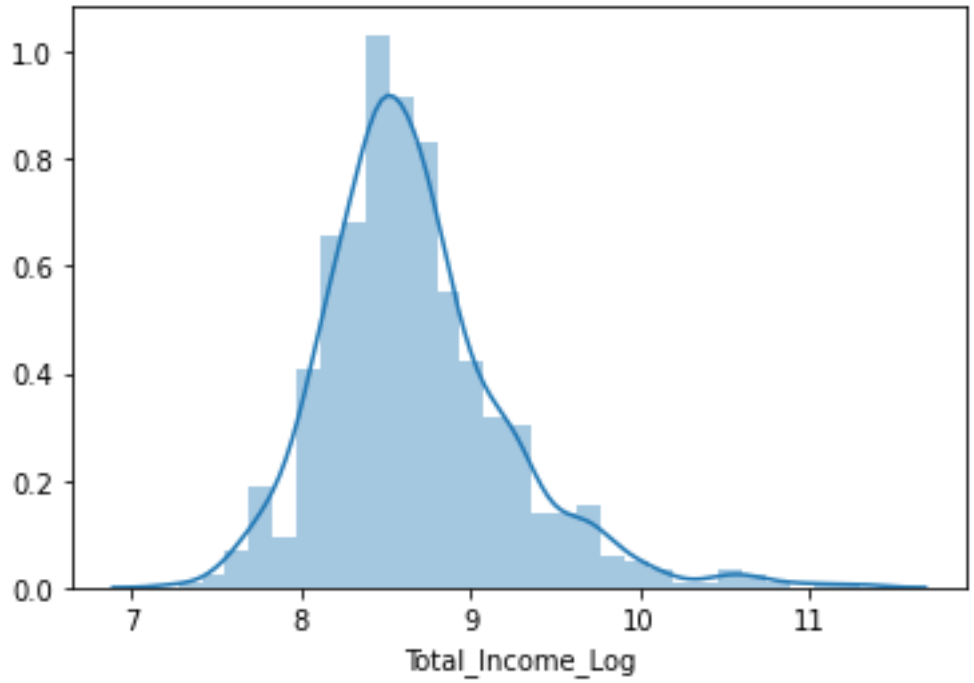
Out[24]:




```
df['Loan_Amount_Term_Log'] =
np.log(df['Loan_Amount_Term']+1)
sns.distplot(df["Loan_Amount_Term_Log"])
```




```
df['Total_Income_Log'] = np.log(df['Total_Income']+1)
sns.distplot(df["Total_Income_Log"])
```

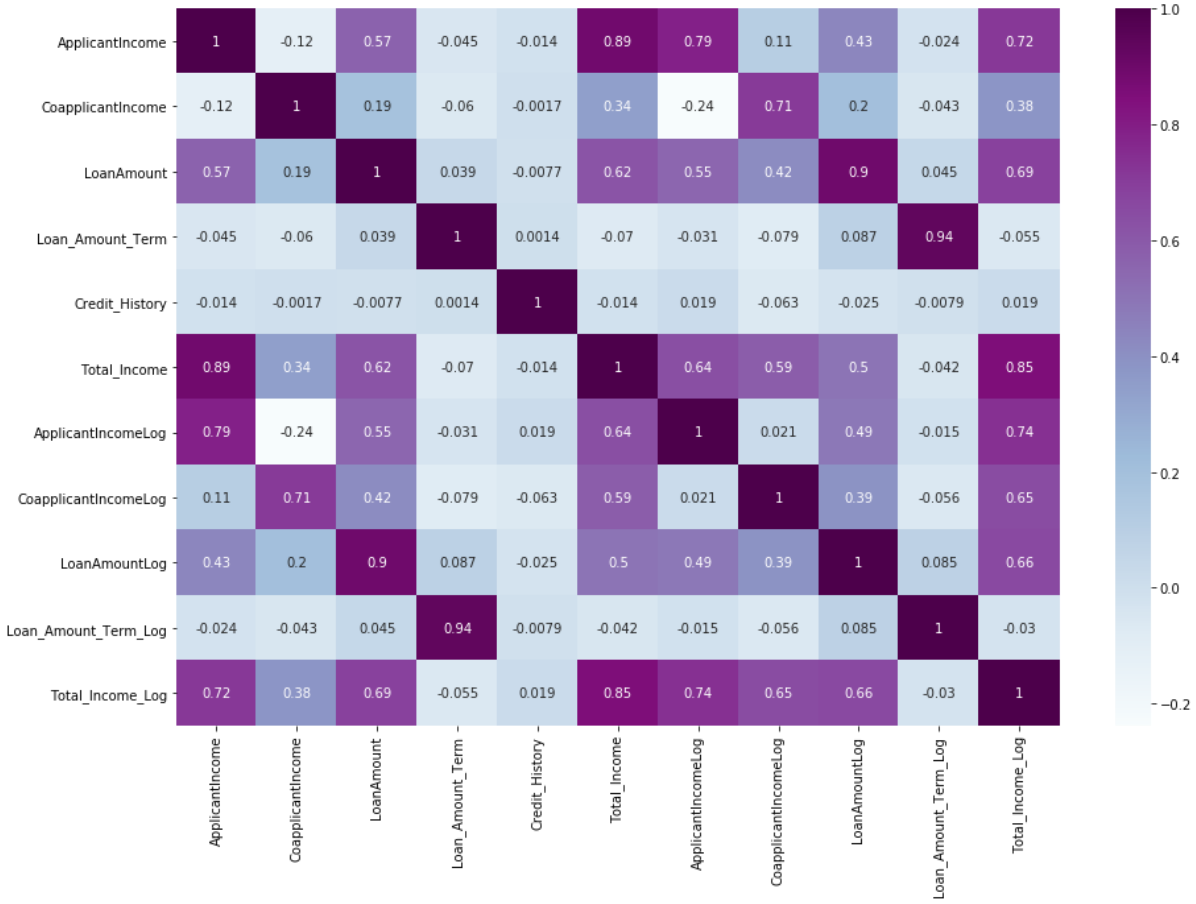


Coorelation Matrix

In [27]:

```
corr = df.corr()
plt.figure(figsize=(15,10))
sns.heatmap(corr, annot = True, cmap="BuPu")
```

Out[27]:



In [28]:

To check the values of the dataset.

```
df.head()
```

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Cr
0	LP001002	Male	No	0	Graduate	No	5849	0.0	146.412162	360.0	
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.000000	360.0	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.000000	360.0	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.000000	360.0	
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.000000	360.0	

Let us drop some unnecessary columns.

```
# drop unnecessary columns cols = ['ApplicantIncome', 'CoapplicantIncome',
"LoanAmount", "Loan_Amount_Term", "Total_Income", 'Loan_ID',
'CoapplicantIncomeLog']

df = df.drop(columns=cols, axis=1)

df.head()
```

	Gender	Married	Dependents	Education	Self_Employed	Credit_History	Property_Area	Loan_Status	ApplicantIncomeLog	LoanAmountLog	Loa
0	Male	No	0	Graduate	No	1.0	Urban	Y	8.674026	4.986426	
1	Male	Yes	1	Graduate	No	1.0	Rural	N	8.430109	4.852030	
2	Male	Yes	0	Graduate	Yes	1.0	Urban	Y	8.006368	4.189655	
3	Male	Yes	0	Not Graduate	No	1.0	Urban	Y	7.856707	4.787492	
4	Male	No	0	Graduate	No	1.0	Urban	Y	8.699515	4.948760	

Label Encoding

We will use label encoding to convert the categorical column into the numerical column.

```
from sklearn.preprocessing import LabelEncoder
 cols = ['Gender', "Married", "Education", 'Self_Employed', "Property_Area", "Loan_Status", "Dependents"]
le = LabelEncoder()
for col in cols:
  df[col] = le.fit_transform(df[col])
```

We access each column from the column list. And for the corresponding column, the 'le.fit_transform()' function will convert the values into numerical then store them into the corresponding column.

```
df.head()
```

	Gender	Married	Dependents	Education	Self_Employed	Credit_History	Property_Area	Loan_Status	ApplicantIncomeLog	LoanAmountLog	Loa
0	1	0	0	0	0	1.0	2	1	8.674026	4.986426	
1	1	1	1	0	0	1.0	0	0	8.430109	4.852030	
2	1	1	0	0	1	1.0	2	1	8.006368	4.189655	
3	1	1	0	1	0	1.0	2	1	7.856707	4.787492	
4	1	0	0	0	0	1.0	2	1	8.699515	4.948760	

- All the values of the dataset are now in numerical format. It will help us to train our model easily.
- For Loan status 1 indicates 'Yes' and 0 indicates 'No'.

Splitting the data for Training and Testing

```
# specify input and output attributes
X = df.drop(columns=['Loan_Status'], axis=1)
 y = df['Loan_Status']
```

Let us now split the data.

```
from sklearn.model_selection import train_test_split x_train, x_test,
y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
```

Model Training

```
# classify function
from sklearn.model_selection import cross_val_score
def classify(model, x, y):
    x_train, x_test, y_train, y_test = train_test_split(X, y,
test_size=0.25, random_state=42)
    model.fit(x_train, y_train)
    print("Accuracy is", model.score(x_test, y_test)*100)
    # cross validation - it is used for better validation of
model
    # eg: cv-5, train-4, test-1
    score = cross_val_score(model, x, y, cv=5)
    print("Cross validation is",np.mean(score)*100)
```

- Here, cross-validation will split the data set into multiple parts.
- For example; **cv=5** means, it will split the data into 5 parts.
- For each iteration, the training will use 4 parts and testing will use 1 part.
- You can change the cross-validation with the common term 3 or 5.

Logistic Regression:

```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
classify(model, X, y)
```

```
Accuracy is 77.27272727272727
Cross validation is 80.79587519830778
```

- Since cross-validation deals with multiple parts, we have to focus on cross-validation percentage, which is an overall accuracy of the model.

Decision Tree:

```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
classify(model, X, y)
```

```
Accuracy is 72.72727272727273
Cross validation is 71.68693812797461
```

- The decision tree does not show good results.

Random Forest:

```
from sklearn.ensemble import
RandomForestClassifier,ExtraTreesClassifier
model = RandomForestClassifier()
classify(model, X, y)
```

```
Accuracy is 78.57142857142857
Cross validation is 75.90957165520888
```

- Random forest shows better results than a Decision tree.

Extra Trees:

```
model = ExtraTreesClassifier()
classify(model, X, y)
```

```
Accuracy is 73.37662337662337
Cross validation is 75.25118984664199
```

this project, Extra tree doesn't show better results than random forest.

Out of all the classifiers, **Logistic Regression** shows a better result in terms of cross-validation. Now let's try to change some hyperparameters to improve the accuracy.

Hyperparameter tuning

We will change some hyperparameters for Random Forest Classifiers.

```
model = RandomForestClassifier(n_estimators=100,
min_samples_split=25, max_depth=7, max_features=1)
classify(model, X, y)
```

```
Accuracy is 76.6233766233766
Cross validation is 79.86677948175568
```

- Generally, we change the parameter with the use of algorithms like Grid Search and Random Search.
- You can also use any algorithm of your convenience.

Confusion Matrix

A confusion matrix is a summary of **prediction results** on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. It gives us insight not only into the **errors** being made by a classifier but more importantly the **types of errors** that are being made.

We will use the Random Forest Model.

```
model = RandomForestClassifier()
model.fit(x_train, y_train)
```

After running the basic default parameters we will plot the confusion matrix.

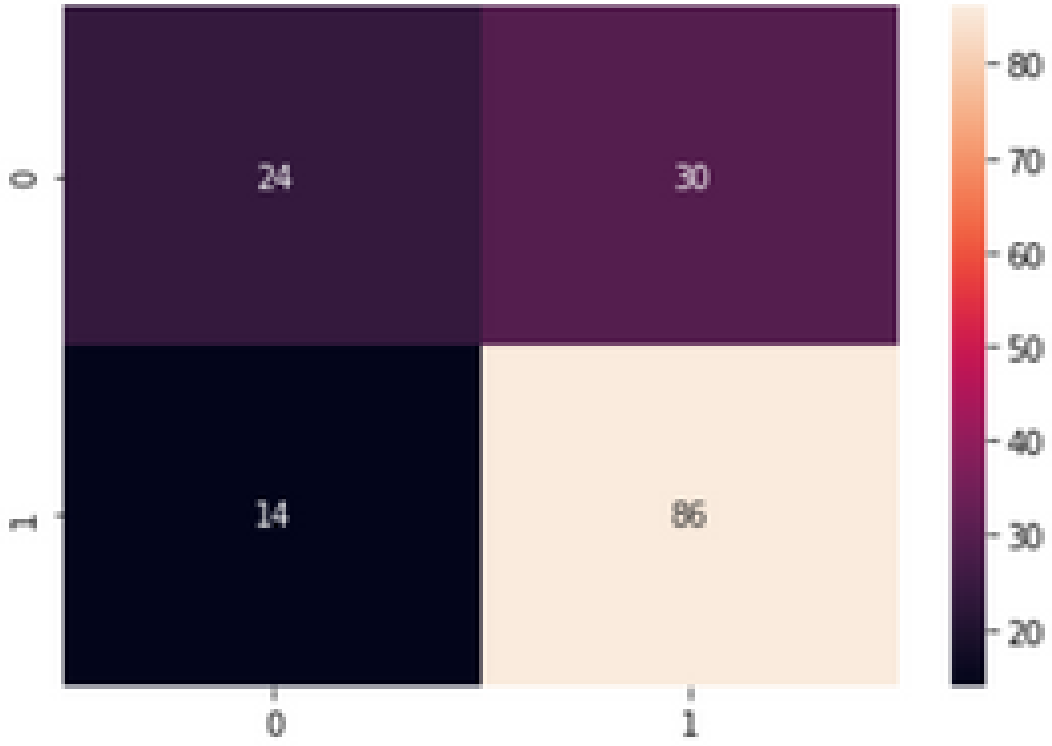
```
from sklearn.metrics import confusion_matrix
y_pred = model.predict(x_test)
cm = confusion_matrix(y_test, y_pred)
cm
```

```
array([[24, 30],
       [14, 86]], dtype=int64)
```

- **y_test** contains the actual values from the dataset.
- **y_predict** contains the predicted values from the model.

To display the confusion matrix in a heat map.

```
sns.heatmap(cm, annot=True)
```



The left side of the heatmap indicates actual values, and the bottom side shows predicted values.

- For actual value '0' there are 24 correct predictions. For actual value '1' there are 86 correct predictions.
- The model has falsely predicted 30 counts for class 0. Therefore, we need to train better for class 0.

Future Scope:

This project is helpful for banking and finance companies because by using this this model they will understand that wheather they should give a lone to a person or not based upon their data.

Conclusion:

From the Exploratory Data Analysis, we could generate insight from the data. How each of the features relates to the target. Also, it can be seen from the evaluation of three models that Logistic Regression performed better than others, Random Forest did better than Decision Tree.

Git Hub LINK-- https://github.com/subhamnandi12345/Lone-Prediction-Analysis

THANK YOU