LOAN PREDICTION ANALYSIS

A mini project report submitted by

AUSTIN VINIL STEPHEN (URK18CS132)

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

under the supervision of

Mr.BL Radhakrishnan, Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

KARUNYA INSTITUTE OF TECHNOLOGY AND SCIENCES
(Declared as Deemed-to-be-under Sec-3 of the UGC Act, 1956)
Karunya Nagar, Coimbatore - 641 114. INDIA

March 2021

Karunya INSTITUTE OF TECHNOLOGY AND SCIENCES
(Declared as Deemed to be University under Sec.3 of the UGC Act, 1956)
A CHRISTIAN MINORITY RESIDENTIAL INSTITUTION
AICTE Approved & NAAC Accredited

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that the project report entitled, "**Loan Prediction Analysis**" is a bonafide record of Mini Project work done during the even semester of the academic year 2020-2021 by

AUSTIN VINIL STEPHEN (Reg. No: URK18CS132)

in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science and Engineering of Karunya Institute of Technology and Sciences.

Submitted for the Viva Voce held on **29th March 2021**

Project Coordinator **Signature of the Guide**

ACKNOWLEDGEMENT

First and foremost, I praise and thank ALMIGTHY GOD whose blessings have bestowed in me the will power and confidence to carry out my project.

I am grateful to our beloved founders Late**. Dr. D.G.S. Dhinakaran, C.A.I.I.B, Ph.D.** and **Dr. Paul Dhinakaran, M.B.A, Ph.D.**, for their love and always remembering us in their prayers.

I extend my thanks to our Vice Chancellor **Dr. P. Mannar Jawahar, Ph.D.** and our Registrar **Dr. Elijah Blessing, M.E., Ph.D.,** for giving me this opportunity to do the project.

I would like to thank **Dr. Prince Arulraj, M.E., Ph.D.,** Dean, School of Engineering and Technology for his direction and invaluable support to complete the same.

I would like to place my heart-felt thanks and gratitude to **Dr. J. Immanuel John Raja, M.E., Ph.D..,** Head of the Department, Computer Science and Engineering for his encouragement and guidance.

I feel it is a pleasure to be indebted to, **Mr. J. Andrew, M.E, (Ph.D.)**, Assistant Professor, Department of Computer Science and Engineering and **Mr. BL Radhakrishnan MTech, (Ph.D.)** for their invaluable support, advice and encouragement.

I also thank all the staff members of the Department for extending their helping hands to make this project a successful one.

I would also like to thank all my friends and my parents who have prayed and helped me during the project work.

ABSTRACT

The loan system has an important role in the modern economy. This system has helped the economy of the country as they are a big support to most industrialists, small businessmen and even for salaried individuals. By giving out loans they facilitate commerce. It acts as a bridge between those who have surplus money and those who are facing money shortage for any of their needs.

When we look at this system through the lens of a banking industry, giving out loans is their main source of profit. The amount of interest they collect from each loan decides their financial success. Therefore, it is very important for a banking company to make their loaning process as undemanding as possible.

As we all know, the traditional way is for the customer to first apply for the loan and then the company will review the eligibility of the customer to receive the loan. This manual procedure is time consuming and very monotonous. The delay caused by this method can diminish the company's potential success.

So, in this mini project with the help of a loan data set collected from Kaggle I will be training a machine learning model to automate the procedure of approving loans. Many deciding factors like credit score, education etc., will be used visualized and used to train the model. With the help of this model the company can be ascertained whether the customer is capable of paying back the loans without the need of extensive manual procedures.

The conclusion we draw from the project is that by using logistic regression, we get the highest cross validation score of around 80.7%. This result is very desirable for the automation of loan processing.

CONTENTS

Acknowledgments 3
Abstract 4

1. Introduction 1
1.1 Introduction 6
1.2 Objectives 6
1.3 Motivation 6
1.4 Overview of the Project 7
1.5 Chapter wise Summary 7

2. Analysis and Design 8
2.1 Dataset Information 8
2.2 Preprocessing the dataset 9
2.3 Exploratory Data Analysis 12

3. Implementation 18
3.1 Creation of new attributes 18
3.1.1 Log Transformation 18
3.2 Correlation Matrix 21
3.3 Label Encoding 22

4. Test results 23
4.1 Train-Test Split 23
4.2 Model Training 24

5. Conclusions and Further Scope 26

References 27

1.INTRODUCTION

1.1 INTRODUCTION

Today many banks/financial companies approve 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. That's where loan prediction comes into play. Loan Prediction is very helpful for employee of banks as well as for the applicant also. The aim of this project is to provide quick, immediate and easy way to choose the deserving applicants. The Loan Prediction System 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. Therefore, with the help of automation productivity and efficiency can be increased overall.

1.2 OBJECTIVES

The objective of this project is to automate the loan eligibility process on the basis of customer details such as Gender, Marital Status, Education, Number of Dependents, Income, Loan Amount, Credit History and others provided by the candidate themselves. With the help of this model the company can identify the customers segments those are eligible for loan amount so that they can specifically target these customers.

1.3 MOTIVATION

Even though the practice of loaning is revolutionary and beneficial for both the bank and its consumer; the execution of the loaning process can be improved. Since faults such as human error, time consumption, incompetent marketing among others hinders the ability of the system to reach its full potential. Resolving these issues would result in increased efficiency, productivity and overall prosperity of the social economy. In the modern day there is a vast availability of data that can be used for analysis with which we can implement such improvements. This led me to devise such an alternative.

1.4 OVERIVEW OF THE PROJECT

We are given a dataset of a bank and we are going to analyze it to build a predictive model that can predict whether the loans will be approved or not using the given input data. We will primarily do the analysis of the data and then we will go on to make the implementations and finally test the models. With the help of these steps, we will be able to determine the most favorable model for predicting loans.

1.5 CHAPTER WISE SUMMARY

This is project is divided into 3 main parts, Analysis and Design, Implementation, and Test results. In Analysis and Design phase we will collect the data and analyze the attributes, then we will go on to do the basic pre processing step by figuring out the missing terms in the data and we will fill it out using the mean and mode function depending on the type of attribute. Finally, we will do Exploratory Data Analysis on both numerical and categorial data and we will analyze the relations. Then we move onto the implementation section, here we will create new attributes to improve the training and analysis, and we use the log function to normalize the attributes and separated corresponding attributes will be created. Then we will create a correlation matrix to find attributes with the highest correlation and drop them as they are redundant and we also drop other unnecessary tables. Finally, in this section we will do Label Encoding where the categorical data is converted into numerical form. Last but not least we have the Test section where we carry out the main operations. Firstly, we will do train-test split and we will assign 25% for testing and the rest for training. Then we will make a reusable classify function so that we can test various models using it. Finally, we will test out the models and look for the model that has the highest cross validation score.

2. ANALYSIS AND DESIGN

2.1 DATASET INFORMATION

A dataset is a collection of data. In the case of tabular data, a dataset corresponds to one or more database tables, where every column of a table represents a particular variable, and each row corresponds to a given record of the dataset.

We are provided with a dataset that has the following attributes

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)

Fig1 – Dataset attributes with description

Out of the 13 attributes given, there are 12 input attributes and our job is to predict the Loan_Status attribute.

Now let us load the first 5 records of the table

```
In [2]: df = pd.read_csv("Loan Prediction Dataset.csv")
        df.head()
```

Out[2]:

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

```
In [2]: df = pd.read_csv("Loan Prediction Dataset.csv")
        df.head()
```

Out[2]:

Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status
0	Graduate	No	5849	0.0	NaN	360.0	1.0	Urban	Y
1	Graduate	No	4583	1508.0	128.0	360.0	1.0	Rural	N
0	Graduate	Yes	3000	0.0	66.0	360.0	1.0	Urban	Y
0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0	Urban	Y
0	Graduate	No	6000	0.0	141.0	360.0	1.0	Urban	Y

Fig2 – First 5 records of the dataset

As we can see, there are categorical data and numerical data in the same dataset and we need to fix it.

2.2 PREPROCESSING THE DATASET

In any Machine Learning process, Data Preprocessing is that step in which the data gets transformed, or Encoded, to bring it to such a state that now the machine can easily parse it. In other words, the features of the data can now be easily interpreted by the algorithm.

Now let us search our dataset for any possible missing values

```
In [5]: # find the null values
        df.isnull().sum()

Out[5]: 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
```

Fig3 – Checking for missing values

Now we can clearly see that there are 7 attributes with missing values (Gender [13], Married [3], Dependents [15], etc.,) and we have to fill up these missing values.

Let's start with the numeric terms

For numeric terms we have to use the mean function, with the help of the mean function we can take the average value of that particular attribute and fill up the missing values of that attribute with the mean we found.

```
In [6]: # 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'].mean())
```

Fig4 – Filling out missing values for numeric terms

As we know 'LoanAmount', 'Loan_Amount_Term', and 'Credit_History' are the numeric attributes with missing values. Hence. we filled the missing values with their average values using the mean function.

Now let's deal with categorical attributes

In this case we can't use mean function because they are categorical values. So, we have to use the mode function. Mode function shows the value that occurs the most

```
In [7]: # 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])
```

Fig5 – Filling out missing values for categorical terms

Here 'Gender', 'Married', 'Dependents' and 'Self_Employed' were the categorical attributes with missing values. Using the code written above we were able to fill the missing values with their modes.

Now let's check if we have missing values any more

```
In [8]: df.isnull().sum()

Out[8]: 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
```

Fig6 – Checking if we still have missing values

We have successfully removed all the missing values and hence, we complete the basic preprocessing steps.

2.3 EXPLORATORY DATA ANALYSIS

Exploratory data analysis is an approach to analyzing data sets to summarize their main characteristics, often using statistical graphics and other data visualization methods

In this section we are going to analyze the data for each relevant attribute.

Let's start with the Gender attribute

```
In [9]: # categorical attributes visualization
        sns.countplot(df['Gender'])

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x20c636b49e8>
```

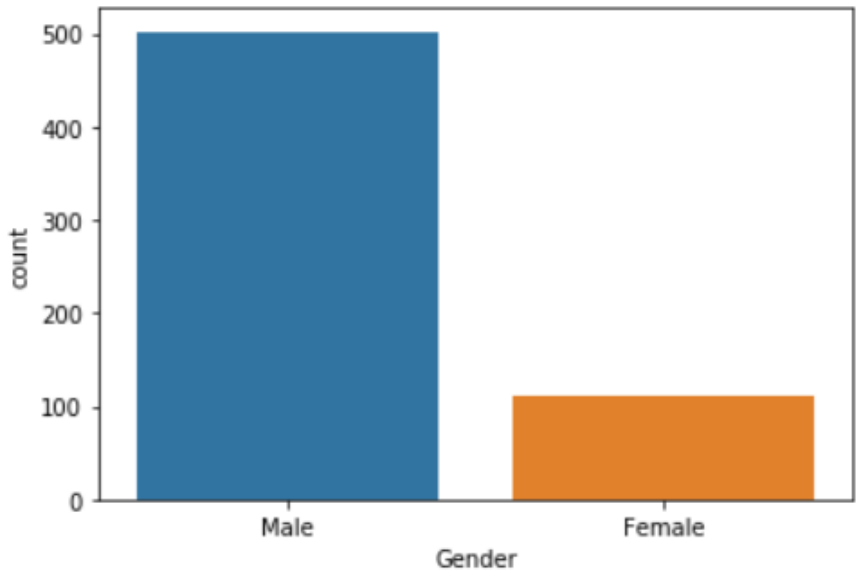


Fig7 – Gender (EDA)

Here we can see that there are more male applicants than female.

Marriage attribute

```
In [10]: sns.countplot(df['Married'])
Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x20c63789c50>
```

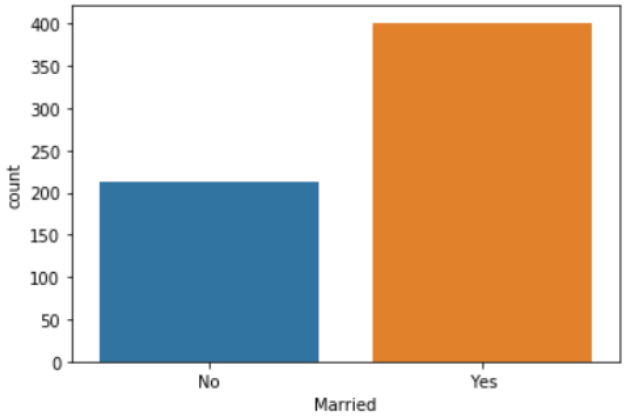


Fig8 – Marriage (EDA)

Here we can see that Married couple are more in number.

Dependents attribute

```
In [11]: sns.countplot(df['Dependents'])
Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x20c637dcfd0>
```

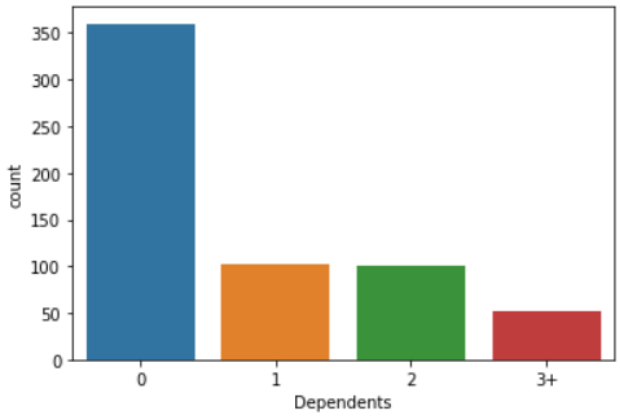


Fig9 – Dependents (EDA)

We can conclude that most of the applicants don't have any dependents.

Education attribute

```
In [12]: sns.countplot(df['Education'])
Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x20c6385bdd8>
```

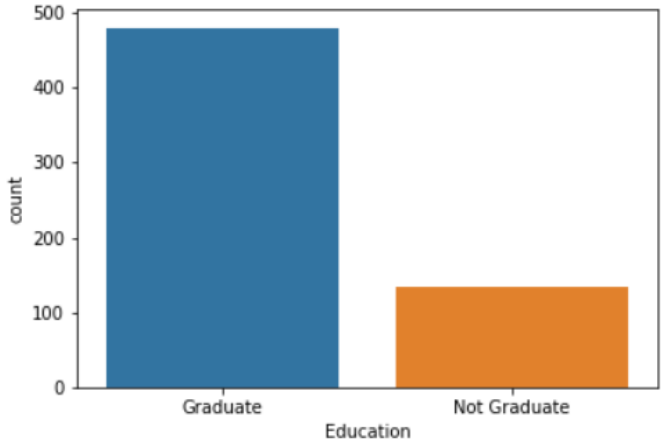


Fig10 – Education (EDA)

Self_Employed attribute

```
In [13]: sns.countplot(df['Self_Employed'])
Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x20c638a4668>
```

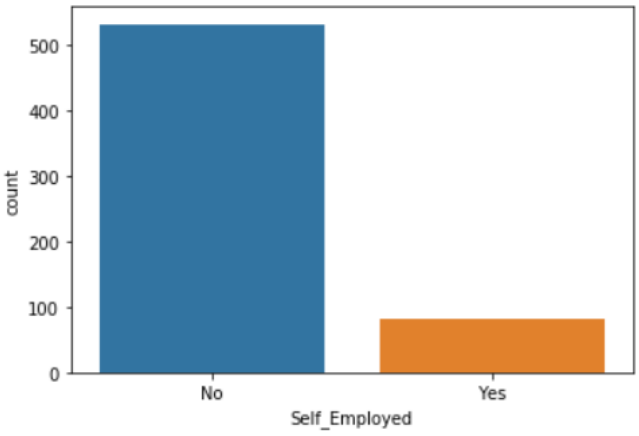


Fig11 – Self_Employed (EDA)

Here we see that there is hardly any self-employed applicants.

Property_Area attribute

```
In [14]: sns.countplot(df['Property_Area'])
Out[14]: <matplotlib.axes._subplots.AxesSubplot at 0x20c638ef6a0>
```

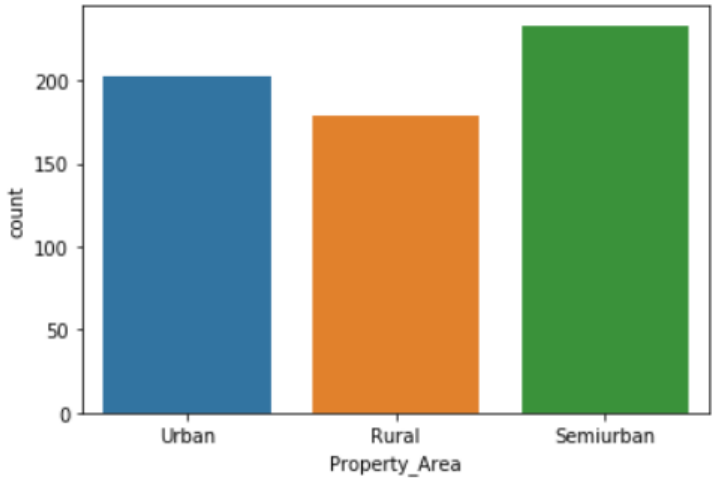


Fig12 – Property_Area (EDA)

This graph does not have much variation as compared to the previous ones.

Loan_Status attribute

```
In [15]: sns.countplot(df['Loan_Status'])
Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x20c6394a128>
```

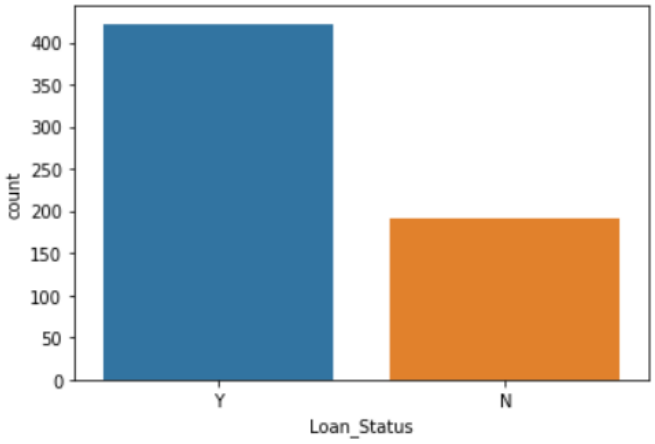


Fig13 – Loan_Status (EDA)

Here we can conclude that only half of the loan application were rejected.

Now let's visualize numerical data

```
In [16]: # numerical attributes visualization
         sns.distplot(df["ApplicantIncome"])
Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x20c63992668>
```

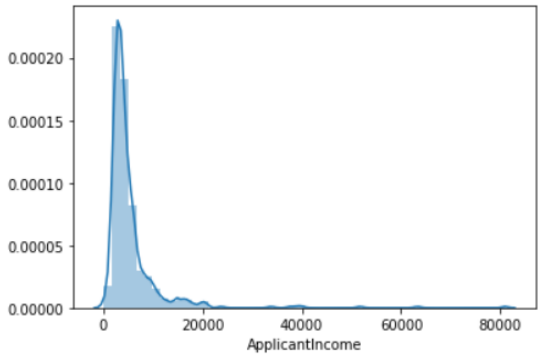
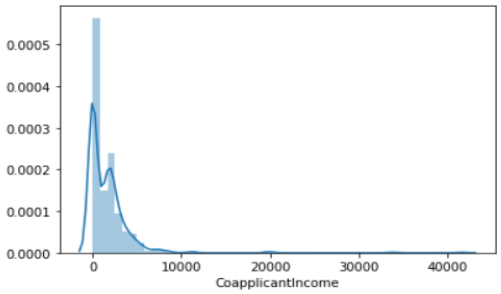


Fig14 – ApplicantIncome (EDA)

```
In [17]: sns.distplot(df["CoapplicantIncome"])
Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0x20c63a91748>
```




```
In [18]: sns.distplot(df["LoanAmount"])
Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x20c641e6a58>
```

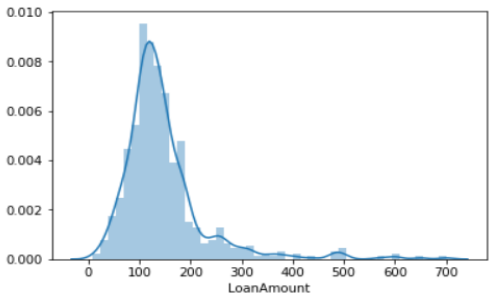
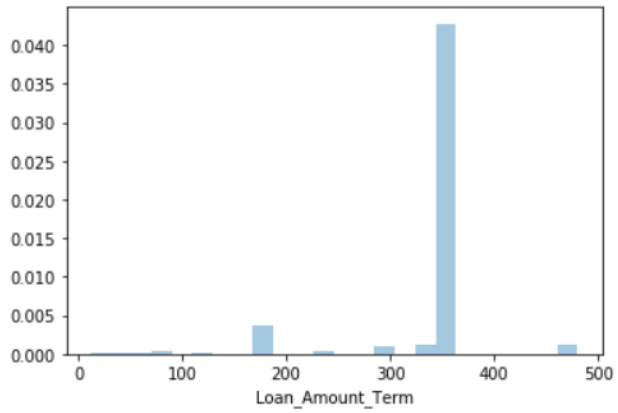


Fig15 – CoapplicantIncome and LoanAmount (EDA)

```
In [19]: sns.distplot(df['Loan_Amount_Term'])
Out[19]: <matplotlib.axes._subplots.AxesSubplot at 0x20c642e55c0>
```




```
In [20]: sns.distplot(df['Credit_History'])
Out[20]: <matplotlib.axes._subplots.AxesSubplot at 0x20c642ddbe0>
```

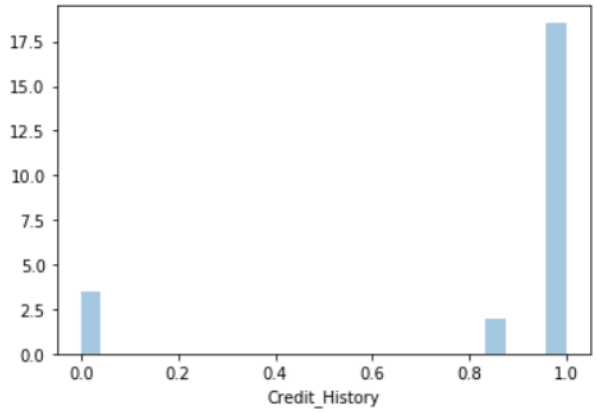


Fig16 – Loan_Amount_Term and Credit_History (EDA)

After looking at the graphs we see that most of them are not normalized.
We shall apply the logarithmic function and normalize them in the following chapter.

3.IMPLEMENTATION

3.1 CREATION OF NEW ATTRIBUTES

We can create a new attribute called Total_Income looking at ApplicantIncome and CoapplicantIncome (there's a high chance that both the incomes are coming from the same household).

```
In [21]: # total income
         df['Total_Income'] = df['ApplicantIncome'] + df['CoapplicantIncome']
         df.head()
```

Out[21]:

Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Property_Area	Loan_Status	Total_Income
Graduate	No	5849	0.0	146.412162	360.0	1.0	Urban	Y	5849.0
Graduate	No	4583	1508.0	128.000000	360.0	1.0	Rural	N	6091.0
Graduate	Yes	3000	0.0	66.000000	360.0	1.0	Urban	Y	3000.0
Not Graduate	No	2583	2358.0	120.000000	360.0	1.0	Urban	Y	4941.0
Graduate	No	6000	0.0	141.000000	360.0	1.0	Urban	Y	6000.0

Fig17 – Total_Income

Now the dataset has a new attribute which is the sum of applicant income and coapplicant income.

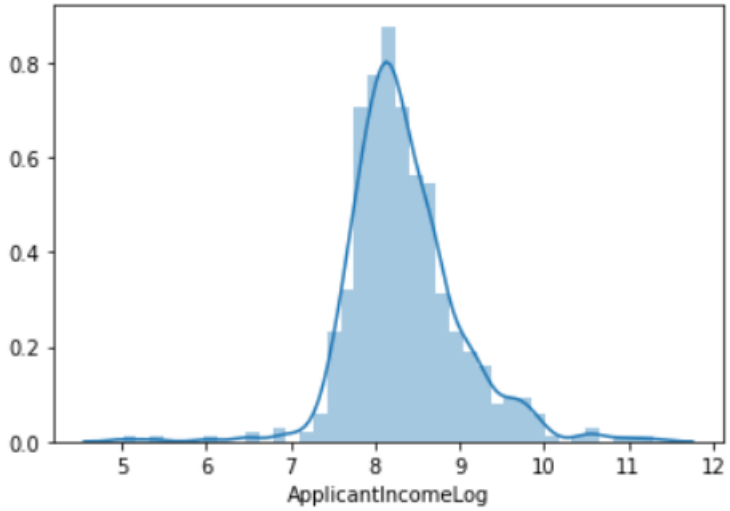
3.1.1 LOG TRANSFORMATION

When we did the visualization of the numerical data attributes, we saw a lot of skewed data. We can make normalize the values with the help of log transformation.

When we normalize the values, we are going to add new attributes that has the log values of the corresponding attributes.

```
In [22]: # apply log transformation to the attribute
         df['ApplicantIncomeLog'] = np.log(df['ApplicantIncome'])
         sns.distplot(df["ApplicantIncomeLog"])

Out[22]: <matplotlib.axes._subplots.AxesSubplot at 0x20c6445ff60>
```




```
In [23]: df['CoapplicantIncomeLog'] = np.log(df['CoapplicantIncome'])
         sns.distplot(df["ApplicantIncomeLog"])

Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x20c64542400>
```

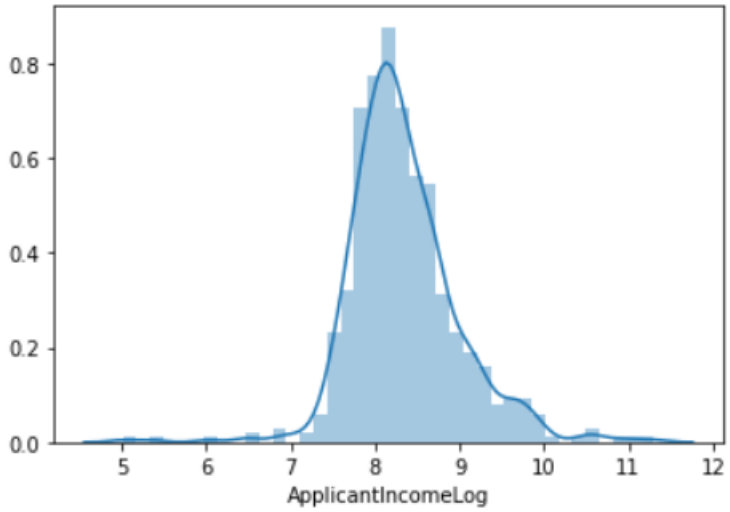
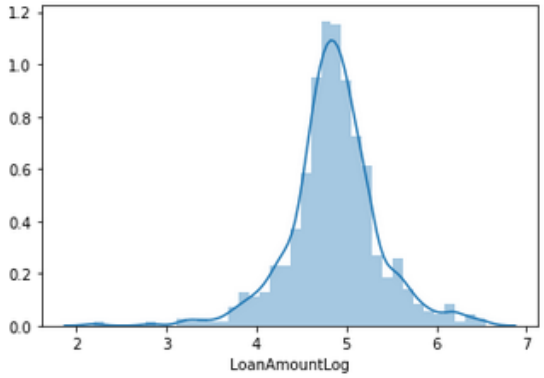


Fig18 – ApplicantIncomeLog and CoapplicantIncomeLog

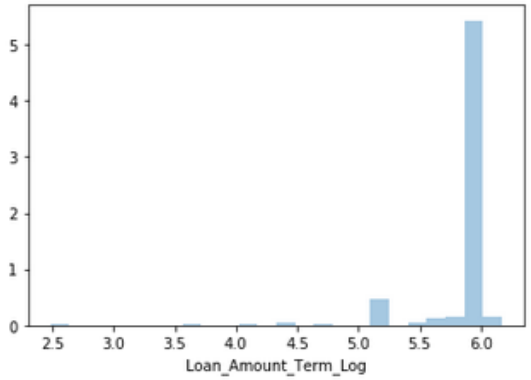
```
In [24]: df['LoanAmountLog'] = np.log(df['LoanAmount'])
         sns.distplot(df["LoanAmountLog"])

Out[24]: <matplotlib.axes._subplots.AxesSubplot at 0x20c64608e48>
```




```
In [25]: df['Loan_Amount_Term_Log'] = np.log(df['Loan_Amount_Term'])
         sns.distplot(df["Loan_Amount_Term_Log"])

Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0x20c646bbd30>
```




```
In [26]: df['Total_Income_Log'] = np.log(df['Total_Income'])
         sns.distplot(df["Total_Income_Log"])

Out[26]: <matplotlib.axes._subplots.AxesSubplot at 0x20c64779898>
```

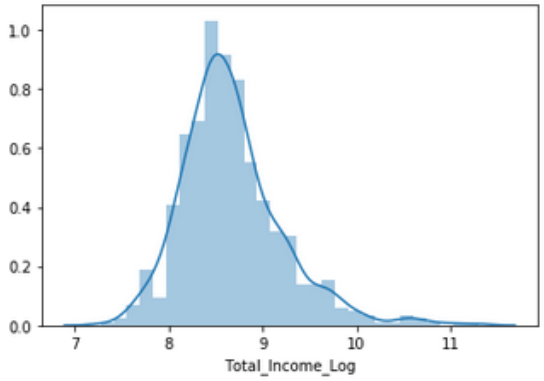


Fig19 – LoanAmountLog and Loan_Amount_Term_Log and Loan_Amount_Term_Log

3.2 CORREATION MATRIX

Here we will use the correlation matrix to find the correlation between all the numerical values.

Our aim here is to find out which pairs have the highest correlation.

```
In [27]: corr = df.corr()
         plt.figure(figsize=(15,10))
         sns.heatmap(corr, annot = True, cmap="BuPu")
```

```
Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x20c6479bf28>
```

Fig20 – <u>Correlation Matrix</u>

Here we can see quite a few values that has high correlation.

For Example., Loan_Amount_Term and Loan_Amount_Term_Log have a very high correlation of 0.94, this implies that they have similar values.
Therefore, we can drop one of the attributes as it can be redundant while training. We shall drop the Loan_Amount_Term table as the log table of it can be more useful for training the model.

Dropping unnecessary columns

As were close to the training stage we have to streamline the columns that were going to use to train.

```
In [29]: # drop unnecessary columns
cols = ['ApplicantIncome', 'CoapplicantIncome', "LoanAmount", "Loan_Amount_Term", "Total_Income", 'Loan_ID', 'CoapplicantIncom
eLog']
df = df.drop(columns=cols, axis=1)
df.head()
```

Out[29]:

	Gender	Married	Dependents	Education	Self_Employed	Credit_History	Property_Area	Loan_Status	ApplicantIncomeLog	LoanAmountLog	Loan_Amoun
0	Male	No	0	Graduate	No	1.0	Urban	Y	8.674026	4.986426	5.886104
1	Male	Yes	1	Graduate	No	1.0	Rural	N	8.430109	4.852030	5.886104
2	Male	Yes	0	Graduate	Yes	1.0	Urban	Y	8.006368	4.189655	5.886104
3	Male	Yes	0	Not Graduate	No	1.0	Urban	Y	7.856707	4.787492	5.886104
4	Male	No	0	Graduate	No	1.0	Urban	Y	8.699515	4.948760	5.886104

Fig21 – Dropping unnecessary columns

After analyzing the correlation matrix, it was decided to drop the numerical tables (ApplicantIncome, CoapplicantIncome, LoanAmount, Loan_Amount_Term) and keep their log columns.

We dropped Loan_ID as it won't be useful for training the model.

CoapplicantIncomeLog was dropped as it has negative infinite values and it could disrupt the training process.

Even though both the coapplicant income tables were dropped their values are not completely gone because we mapped it to the total income attribute.

3.3 LABEL ENCODING

In our data set we have a lot of categorical data that can't be read by the machine. Hence. we use Label Encoding to convert into numerical form so we will be able to train the model using this data.

```
In [30]: 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])
```

```
In [31]: df.head()
```

Out[31]:

	Gender	Married	Dependents	Education	Self_Employed	Credit_History	Property_Area	Loan_Status	ApplicantIncomeLog	LoanAmountLog	Loan_Amoun
0	1	0	0	0	0	1.0	2	1	8.674026	4.986426	5.886104
1	1	1	1	0	0	1.0	0	0	8.430109	4.852030	5.886104
2	1	1	0	0	1	1.0	2	1	8.006368	4.189655	5.886104
3	1	1	0	1	0	1.0	2	1	7.856707	4.787492	5.886104
4	1	0	0	0	0	1.0	2	1	8.699515	4.948760	5.886104

Fig22 – Label Encoding

After applying Label Encoding all the categorical values got converted into numeric and now training can be done easily.

4.TEST RESULTS

4.1 TRAIN-TEST SPLIT

Before we do the train-test split, let us assign the input and output attributes.

X will be input so well drop the Loan_Status attribute as it's the output attribute.
Y will be output and it will only have the Loan_Status attribute.

```
In [32]: # specify input and output attributes
         X = df.drop(columns=['Loan_Status'], axis=1)
         y = df['Loan_Status']
```

Fig23 – Assigning input and output

Now let's implement the train-test split

```
In [33]: 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)
```

Fig24 – Train-Test Split

Here the test size is given for 25%.
So, we'll be training with 75% of the data.

A hyperparameter called random_state is used, we use this to get the same output after each execution.

4.2 MODEL TRAINING

In this step we're going to provide ML algorithms with the training data we have to make an ML model.

With the help of the target attribute in the training data the learning algorithm finds patterns in the training data that map the input data attributes to the target, and it outputs an ML model that captures these patterns.

Let us make a function so that we can reuse it for different models

```
In [34]: # 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)
```

Fig25 – Classify function

Here we will use cross validation as it is the better form of validation. Cross validation will split that dataset into multiple parts (depends on the value we mention) and iterate with different parts each time.

As the function is ready let us try out different models and check the output

LogisticRegression

```
In [35]: from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
classify(model, X, y)

Accuracy is 77.27272727272727
Cross validation is 80.79587519830778
```

DecisionTreeClassifier

```
In [36]: from sklearn.tree import DecisionTreeClassifier
         model = DecisionTreeClassifier()
         classify(model, X, y)

         Accuracy is 72.72727272727273
         Cross validation is 71.68693812797461
```

RandomForestClassifier

```
In [37]: from sklearn.ensemble import RandomForestClassifier,ExtraTreesClassifier
         model = RandomForestClassifier()
         classify(model, X, y)

         Accuracy is 78.57142857142857
         Cross validation is 75.90957165520888
```

ExtraTreesClassifier

```
In [38]: model = ExtraTreesClassifier()
         classify(model, X, y)

         Accuracy is 73.37662337662337
         Cross validation is 75.25118984664199
```

After analyzing the Cross Validation scores of the models after training we can say that LogisticRegression performs the best with 80.7% accuracy.

5. CONCLUSIONS AND FURTHER SCOPE

The analytical process started from data cleaning and processing, missing value imputation, then exploratory analysis and finally model building and evaluation. A model with an accuracy of 80.7% was successfully trained. The previous works of the project judged the model based on its accuracy score, but here in my project the cross validation score is used as it does a more in-depth analysis of the model. This makes it easier for us to identify the ideal model for loan prediction analysis. Even though 80.7% is an impressive score, it is not ideal for real world usage. With the help of hyperparameter tuning we will be able to improve the accuracy and make it ready for real world usage.

This is a project has a major scope for further development. As we all know millions of data is being generated every second and with the help of Artificial Intelligence and Deep Learning, we will be able to utilize the increasing mass of data to improve the accuracy of the model as close to the ideal form as possible. With the help of this close to ideal models the real-life implementations will have very desirable results

Using this model companies will be able to target the audience they have to advertise their products to. This model also has the potential to prevent the losses a company bears due to human errors while sanctioning a loan. Most importantly it's going to make everything faster and more convenient when it comes to applying for a loan and this can attract a greater audience. And as we know loans have helped to boost the economy of our country, this model can really improve the fluidity in our commerce.

REFERENCES

- **https://www.kaggle.com/altruistdelhite04/loan-prediction-problem-dataset**
- **https://datahack.analyticsvidhya.com/contest/practice-problem-loan-prediction-iii/**