# Machine Learning based model for Loan Amount Prediction and Distribution

Hitesh K. Sharma [1], Tanupriya Choudhury [2], Prashant Ahlawat[3], Sachi N. Mohanty [4], Sarika Jain[5]

*[1,2] University of Petroleum & Energy Studies (UPES), Dehradun- 248007,India.**
*[3]Dept. of Computer Science, Chandigarh University, Punjab 140413,India*
*[4]!Department of Computer Science, Singidunum University, Serbia and and School of Computer Science & Engineering, VIT-AP University, Amaravati, Andhra Pradesh, India.*
*[5]Dept. of Computer Application, National Institute of Technology,Kurukshetra,136119, Haryana, India.*

! authors contributed equally and all are the first author.
*are the corresponding authors- Tanupriya Choudhury and Hitesh Kumar Sharma.

**Abstract**
With the enhancement of the technology, expand of businesses and thoughts more and more people are applying for the loans. Both for their personal use and for business use. But due to the limited amount of assets the bank cannot grant loans to each and every person. Finding out those right people is a typical and time-consuming process. Banks desire to deliver the loan to an individual who can be recompensate the loan on time and can afford maximum profit to the bank. So there is a need of a system which could do this analysis and save the banks time and resources. This can be done using Machine Learning. The objective of this paper is to create a more accurate loan prediction model using machine learning to reduce the risk behind selecting of appropriate people for the loan. For this we'll mine the previous records of the people to whom the bank has granted loan. Using these records variables and bank loan rules we will train a machine learning model which will predict that a person is eligible for loan or not. We will use sklearn for our model and train_test_split for splitting the data set into train dataset and test dataset. Here we are going to use various models like Logistic Regression, Decision Tree(DT) and Random Forest(RF) so as to fetch more accurate results as the given problem is a supervised classification problem.

**Keywords**
Machine Learning, Decision Tree, Logistic Regression, Random Forest, Loan, Mine, Train, Prediction.

## 1. Introduction

Maximum profit of the bank comes from lending loans to the people. So distribution of loans is a very important part for every bank. Loan needs to be given to the right person otherwise the bank can face financial trouble and lack of profits. Banks aim to invest their assets in safe hands and from where they will get maximum interest. There are various factors that banks investigate before lending a loan

ACI'22: Workshop on Advances in Computation Intelligence, its Concepts & Applications at ISIC 2022, May 17-19, Savannah, United States
EMAIL: hkshitesh@gmail.com (A.1); tanupriya1986@gmail.com*, (A.2); sachinandan09@gmail.com (A.3); anilgrcse@gmail.com (A.4); rohinaruna@gmail.com (A.5)
ORCID: 0000-0001-6816-0324 (A. 1); 0000-0002-9826-2759 (A. 2); 0000-0002-4939-0797 (A. 3); 0000-0002-4256-5873 (A. 4); 0000-0001-5809-7317 (A. 5)

such as whether a person will be able to repay the loan, what is his financial condition, for what purpose he/she wants to have the loan, etc. Many banks undergo regress processes for choosing a right applicant for loan even though there is still no assurance that the applicant who is selected is the deserving applicant from all others candidates. This wastes the bank as well as the customers time. Through this system we will be able to predict whether the applicant is safe to lend a loan or not to a level of accuracy using the machine learning technique. As the whole process is automated no one will be able to alter the results, it will save the banks and customers time and will lead to quick paperwork. Instead of waiting for a few days to get the whole process done. This machine learning model will really help bank employees and the customers. There are various factors on which this system makes the prediction, but sometimes in real life only one strong factor could be enough for granting the loan to the person. The data of previous records of customers will be mined so as to train our model. Data mining is the process to extract useful information from the large dataset. Classification, clustering and association are the types of data mining. Classification is the main type. There are various classification techniques such as decision tree, neural network, , support vector machine and logistic regression, k-nearest neighbour etc. In machine learning we need two types of datasets: training dataset which will be used to train the model and other is test dataset which will be used to test the model. These both datasets are taken from one dataset. In this paper we will use the train_test_split function of model_selection which will split the data into training dataset and test dataset. Tree representation is used to solve the problem in which each and every leaf node represents a class label and internal nodes represent the attributes. Any boolean function on discrete attributes can be represented using a decision tree. Random forest, which also fits to the supervised learning category. Both classification and regression problems can be solved by this. The concept of ensemble learning is used, which in turn is the process of combining multiple classifiers in order to solve a complex problem and also to progress the performance. It basically produces decision tree set from a random subset selected from the training set and then gathers the votes from different decision trees to make the final prediction. Logistic Regression, it is a supervised classification Algorithm. Logical regression forecasts the categorical dependent variables. Therefore, the outcome should be a categorical or discrete value. It is much similar to Linear regression. The core objective of this paper is to create a less complex system for prediction of loan model. This model has been implemented in python language by using Google Colab software, pandas library for data manipulation and seaborn library for data visualization.

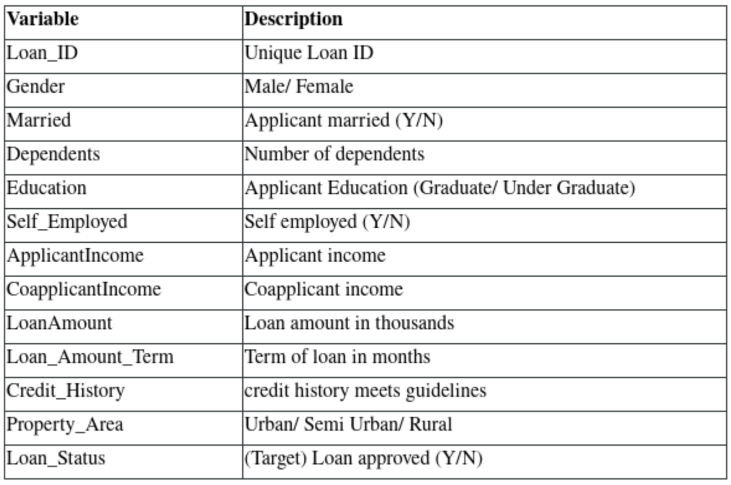
## 2. Literature Review

We studied various research papers on loan prediction models. [1] A research paper by G. Arutjothi and Dr. C. Senthamaria explained that predicting credit defaulters is a complex task so there is a need for a machine learning model for this to save time and resources. Using the R software they have proposed that the combination of Min-Max normalization and K- Nearest Neighbor (K-NN) classifier will be good to accurately predict loan approvals. [2] Aboobyda and Tagir from University of Khartoum, Sudan used j48, bayesNet and naiveBayes algorithms for loan prediction and concluded that j48 would be best for the accurate prediction of credit approvals. They have used Weka application for implementation and testing of the model and compared the results of all the three algorithms.[3] A research paper by Kumar Arun, Garg Ishan and Kaur Sanmeet explained the use of various classification models such as Random Forest, SVM, LM, Nnet and ADB for the prediction of the loan approvals. [4] Glorfeld and Hardgrave had projected a high-performance model with optimum design for the use of neural network thus estimating the credit value of the applications of loans. 75% of the loan applicants were correctly predicted by their designed model. [5] Andy Liaw and Matthew Wiener in their research paper told about the classification and regression by Random Forest. [6] Stephan Dreiseitl and Lucila Ohno-Machado told about the artificial neural network classification and logistic regression models, How logical regression is useful in building various systems for predictions. Machine learning predictive models are also a good choice for loan predictions in this market scenario[7][8][9].

## 3. Proposed Model

### 3.1.1. Data analysis

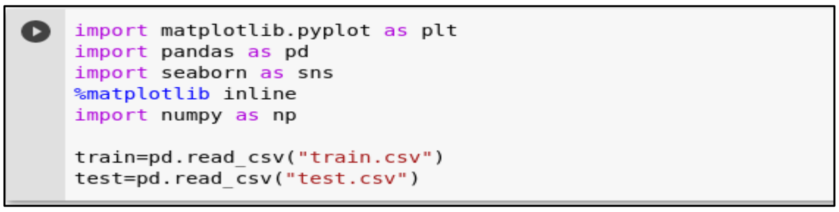
Firstly, we will be doing exploratory data analysis then preprocessing and then finally we will be testing different machine learning models on this data. The data set (Fig. 1) that we are using consists of the following columns.



| 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 | (Target) Loan approved (Y/N) |

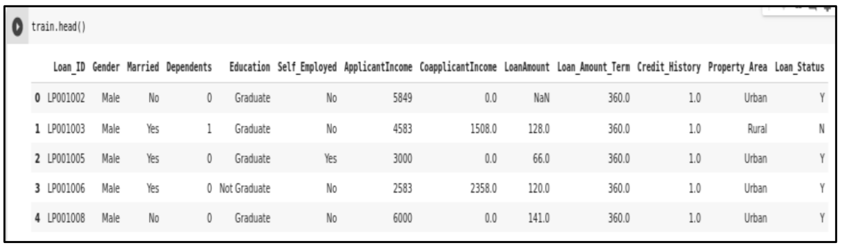
**Figure 1**: Data set columns

We will import (Fig. 2) certain important libraries into the code such as seaborne for the visualization purpose and pandas library for data manipulation purpose. We will also be loading the dataset in the code. Using the head function (Fig. 3,4) we can see a few rows of our dataset.



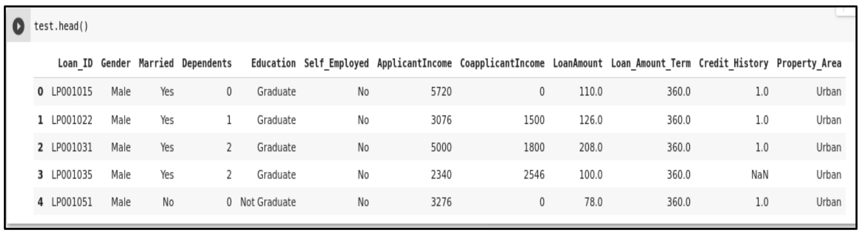
```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
%matplotlib inline
import numpy as np

train=pd.read_csv("train.csv")
test=pd.read_csv("test.csv")
```

**Figure 2:** Importing important libraries



```
train.head()

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

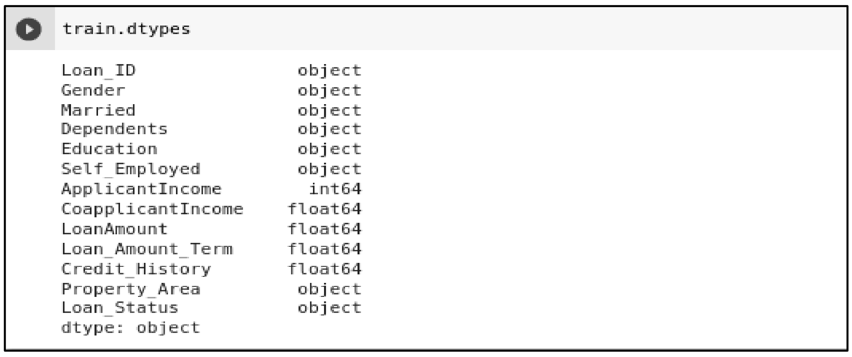
**Figure 3:** Head view of the train data set



```
test.head()

   Loan_ID Gender Married Dependents    Education Self_Employed ApplicantIncome CoapplicantIncome LoanAmount Loan_Amount_Term Credit_History Property_Area
0 LP001015   Male     Yes          0     Graduate            No            5720                 0      110.0            360.0            1.0         Urban
1 LP001022   Male     Yes          1     Graduate            No            3076              1500      126.0            360.0            1.0         Urban
2 LP001031   Male     Yes          2     Graduate            No            5000              1800      208.0            360.0            1.0         Urban
3 LP001035   Male     Yes          2     Graduate            No            2340              2546      100.0            360.0            NaN         Urban
4 LP001051   Male      No          0 Not Graduate            No            3276                 0       78.0            360.0            1.0         Urban
```

**Figure 4**:  Head view of the test data set

Let’s see the data types of all the columns of our train dataset (Fig. 5).



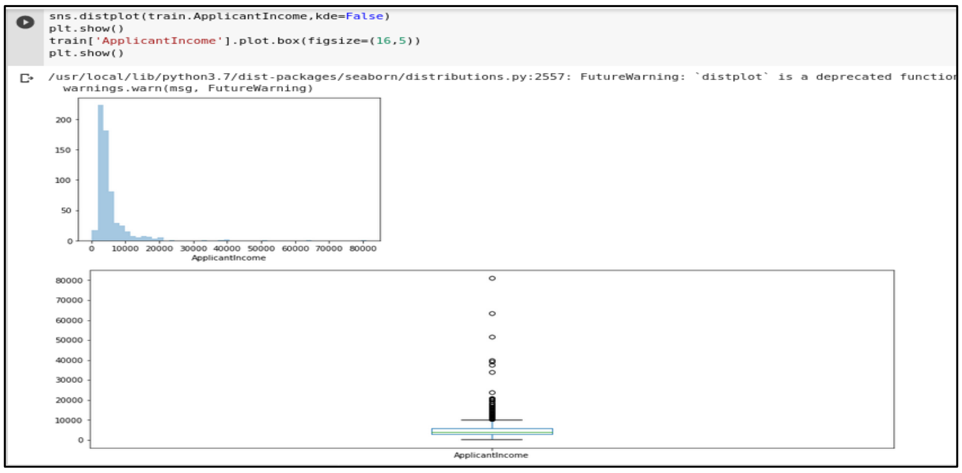
```
train.dtypes

Loan_ID              object
Gender               object
Married              object
Dependents           object
Education            object
Self_Employed        object
ApplicantIncome       int64
CoapplicantIncome   float64
LoanAmount          float64
Loan_Amount_Term    float64
Credit_History      float64
Property_Area        object
Loan_Status          object
dtype: object
```

**Figure 5**: Data types of all the variables in the data

As shown above there are 3 data types:

1. object: This means that the variables are categorical
2. int64: This shows the integer variables
3. float64: This data type shows variables have decimal values

Now let's study about the distribution of numerical variables. Let's see the applicant income and loan amount. We will be doing this with the help seaborn's visualisation.

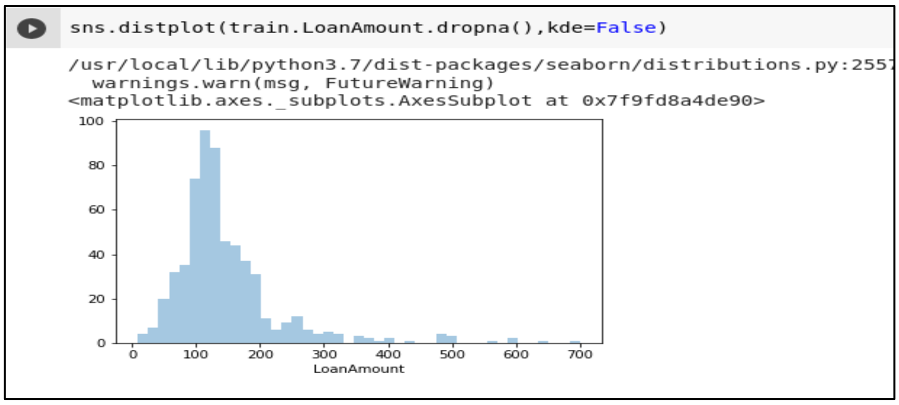


```
sns.distplot(train.ApplicantIncome,kde=False)
plt.show()
train['ApplicantIncome'].plot.box(figsize=(16,5))
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557: FutureWarning: `distplot` is a deprecated functio
  warnings.warn(msg, FutureWarning)
```

**Figure 6:** Graph for Applicant Income

By seeing the graph (Fig. 6) we can say that distribution is suddenly changing its position and also has few deviations. This may be due to the missing values in the dataset. So, we can drop those missing values and again plot the graph with loan amount. We can achieve this with the dropna function (Fig. 7).

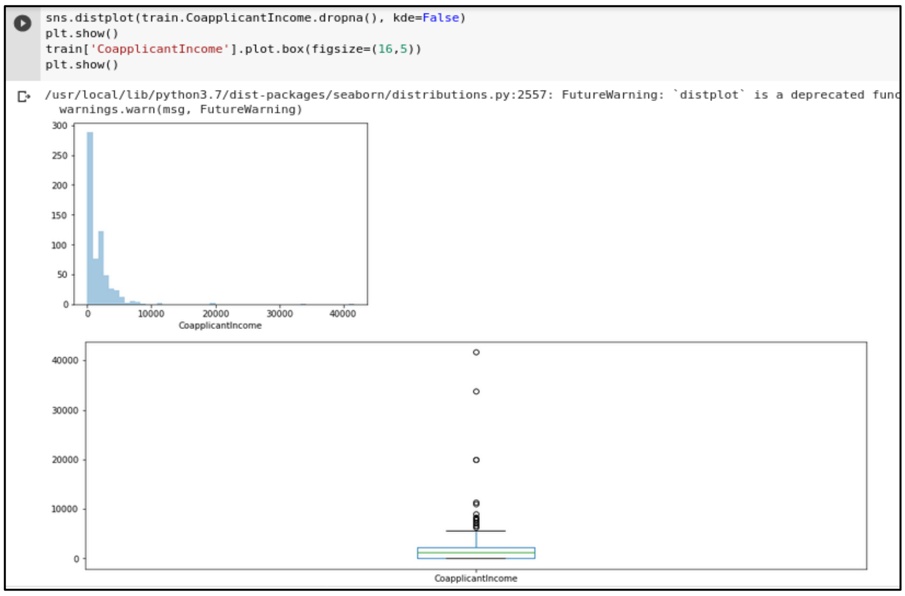


```
sns.distplot(train.LoanAmount.dropna(),kde=False)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f9fd8a4de90>
```

**Figure 7:** Graph for the Loan Amount variable

Now let’s see for the Co-applicant income distribution (Fig 8).

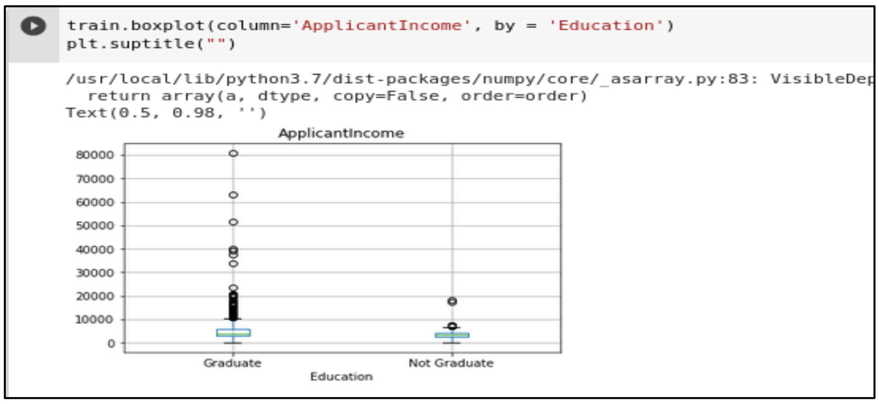


```
sns.distplot(train.CoapplicantIncome.dropna(), kde=False)
plt.show()
train['CoapplicantIncome'].plot.box(figsize=(16,5))
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557: FutureWarning: `distplot` is a deprecated func
  warnings.warn(msg, FutureWarning)
```

**Figure 6:** Graph for the Co Applicant Income

It looks similar to the applicant income distribution. As people who are more educated should be having higher income than people who are not. So let's plot the graph with education level and income (Fig 9).

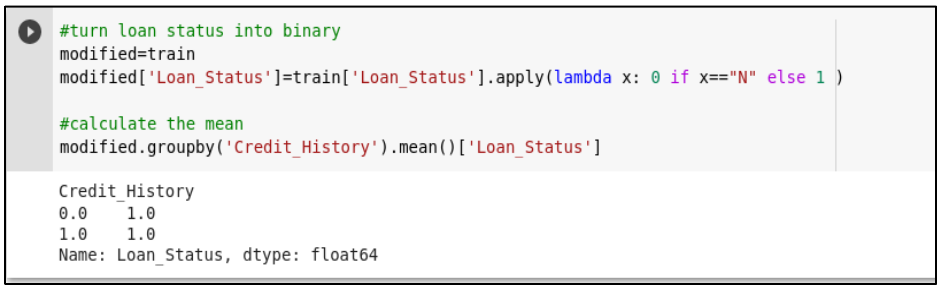


```
train.boxplot(column='ApplicantIncome', by = 'Education')
plt.suptitle("")
```

```
/usr/local/lib/python3.7/dist-packages/numpy/core/_asarray.py:83: VisibleDep
  return array(a, dtype, copy=False, order=order)
Text(0.5, 0.98, '')
```

**Figure 7:** Graph of Applicant Income by Education

From the above graph we can see that the graduate people are having more deviation which shows that the people with high income are well educated (Fig. 9). Loan history can also be another interesting variable which can affect the loan prediction. We can calculate the mean of each value of loan history by turning loan status to 0 or 1. Values closer to 1 will indicate the high loan success rates.



```
#turn loan status into binary
modified=train
modified['Loan_Status']=train['Loan_Status'].apply(lambda x: 0 if x=="N" else 1 )

#calculate the mean
modified.groupby('Credit_History').mean()['Loan_Status']
```

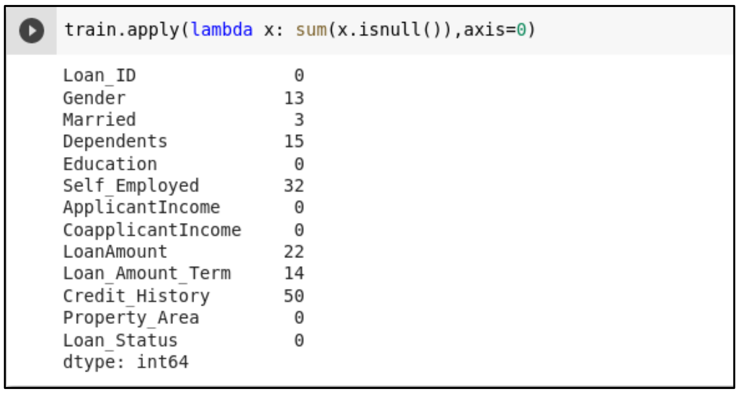
```
Credit_History
0.0    1.0
1.0    1.0
Name: Loan_Status, dtype: float64
```

**Figure 8:** Mean for Loan History

The results tell that the loan history variable will play an important role in the loan prediction in our model (Fig. 10).

### 3.1.2. Missing data values & processing the data:

The dataset that we are using may or may not have missing value. But from the above graphs we can say that our dataset is having some missing values. Let's check how many missing values (Fig. 11) are there for each variable in our dataset.

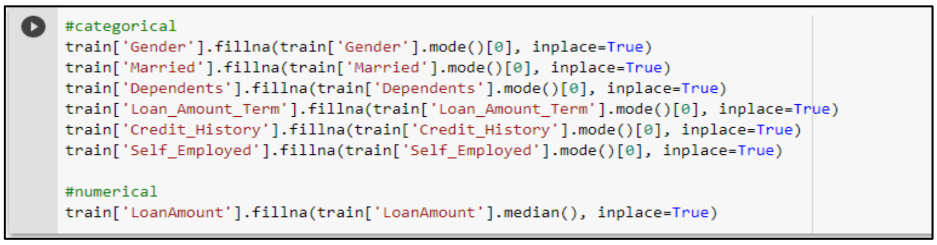


```
train.apply(lambda x: sum(x.isnull()),axis=0)
```

```
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
```

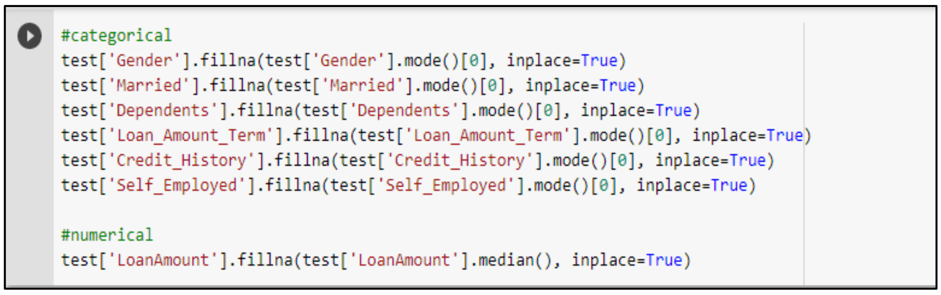
**Figure 11:** Checking for missing values in dataset

One solution for categorical values is that we can fill the missing values (Fig. 12,13) with the mode, which means filling the values with the highest frequency. For the numerical values we can use any mean or median, but as we have seen above there are outliers so using medium will be a proper approach in fill all the missing numerical values (Fig. 14).



```
#categorical
train['Gender'].fillna(train['Gender'].mode()[0], inplace=True)
train['Married'].fillna(train['Married'].mode()[0], inplace=True)
train['Dependents'].fillna(train['Dependents'].mode()[0], inplace=True)
train['Loan_Amount_Term'].fillna(train['Loan_Amount_Term'].mode()[0], inplace=True)
train['Credit_History'].fillna(train['Credit_History'].mode()[0], inplace=True)
train['Self_Employed'].fillna(train['Self_Employed'].mode()[0], inplace=True)

#numerical
train['LoanAmount'].fillna(train['LoanAmount'].median(), inplace=True)
```

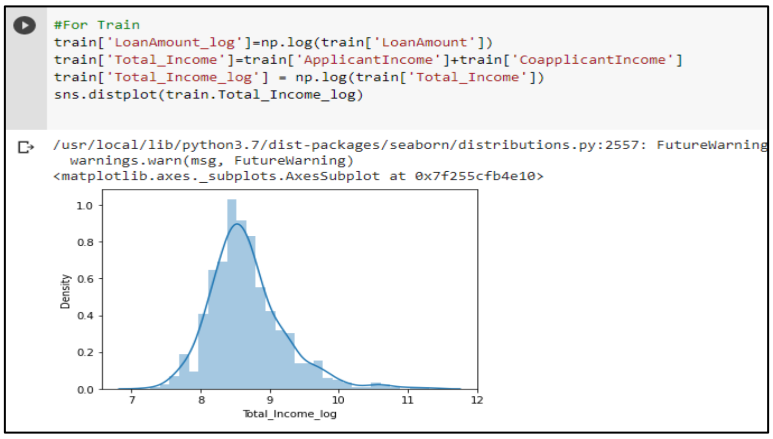
**Figure 12:** Fixing the missing values in train dataset



```
#categorical
test['Gender'].fillna(test['Gender'].mode()[0], inplace=True)
test['Married'].fillna(test['Married'].mode()[0], inplace=True)
test['Dependents'].fillna(test['Dependents'].mode()[0], inplace=True)
test['Loan_Amount_Term'].fillna(test['Loan_Amount_Term'].mode()[0], inplace=True)
test['Credit_History'].fillna(test['Credit_History'].mode()[0], inplace=True)
test['Self_Employed'].fillna(test['Self_Employed'].mode()[0], inplace=True)

#numerical
test['LoanAmount'].fillna(test['LoanAmount'].median(), inplace=True)
```

**Figure 13:** Fixing the missing values in test dataset

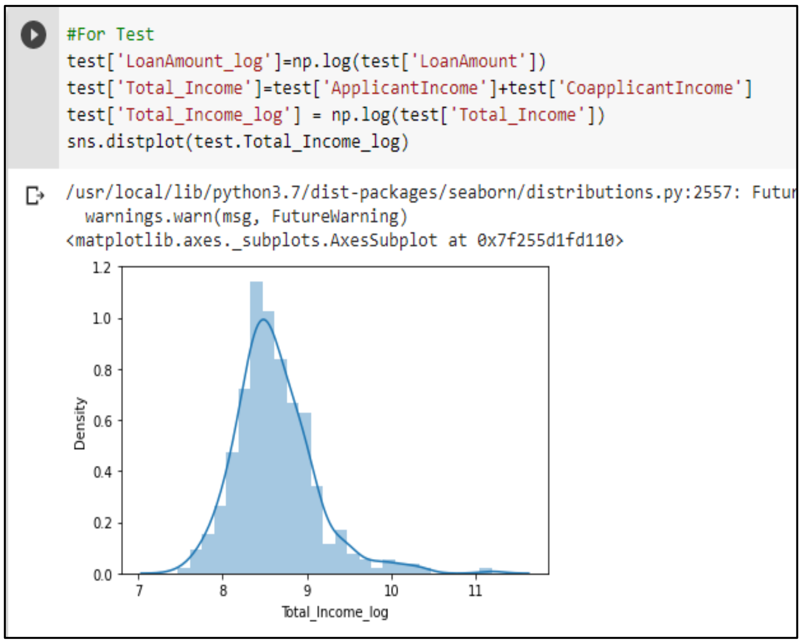
Now let's work on the deviations. We can remove them by log transformation where we will nullify their effect. We will also combine applicant income and co-applicant income to total income column.



```
#For Train
train['LoanAmount_log']=np.log(train['LoanAmount'])
train['Total_Income']=train['ApplicantIncome']+train['CoapplicantIncome']
train['Total_Income_log'] = np.log(train['Total_Income'])
sns.distplot(train.Total_Income_log)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557: FutureWarning
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f255cfb4e10>
```

**Figure 14:** Fixing the outliers for train dataset

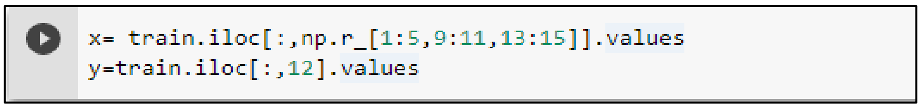


```
#For Test
test['LoanAmount_log']=np.log(test['LoanAmount'])
test['Total_Income']=test['ApplicantIncome']+test['CoapplicantIncome']
test['Total_Income_log'] = np.log(test['Total_Income'])
sns.distplot(test.Total_Income_log)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2557: Futur
  warnings.warn(msg, FutureWarning)
<matplotlib.axes._subplots.AxesSubplot at 0x7f255d1fd110>
```

**Figure 15:** Fixing the outliers for test dataset

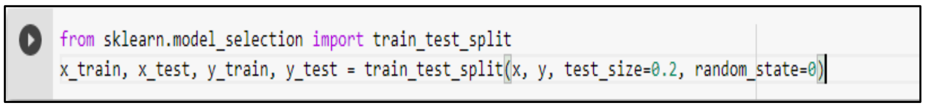
The above graph (Fig. 15) is a histogram of total income log and it seems to be much closer to normal distribution. We will divide our dataset in dependent and independent variables. X will show independent variables and y will show all the dependent variables (Fig. 16).



```
x= train.iloc[:,np.r_[1:5,9:11,13:15]].values
y=train.iloc[:,12].values
```

**Figure 16:** Dividing the train dataset into dependent and independent variables.

Now we will split our dataset into train dataset and test dataset using the train_test_split (Fig. 17).

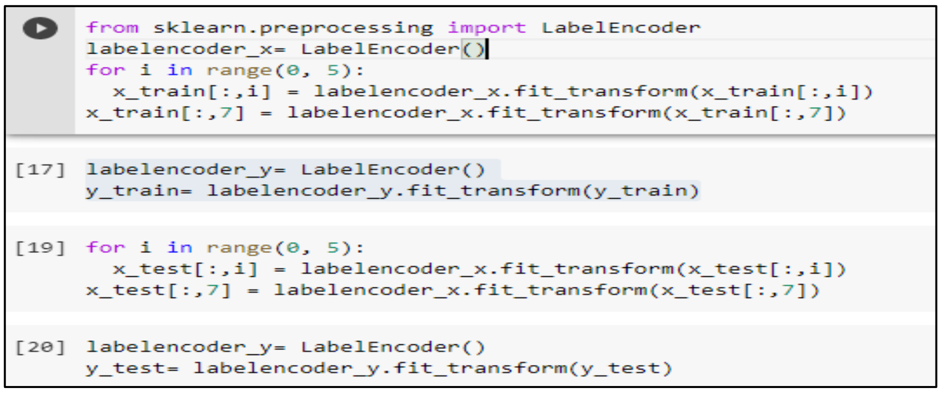


```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)
```

**Figure 17:** Splitting the train dataset

## 3.1.3. Creating the model & Testing:

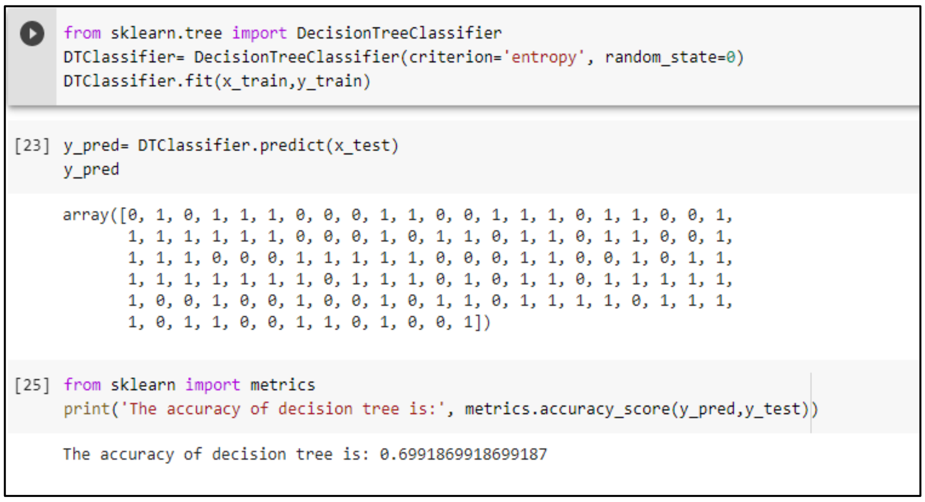
For creating our model we will be using sklearn. But before creating, we will need to change all the categorical variables to numbers. We can do this easily by using LabelEncoder which is present in sklearn.



```
from sklearn.preprocessing import LabelEncoder
labelencoder_x= LabelEncoder()
for i in range(0, 5):
  x_train[:,i] = labelencoder_x.fit_transform(x_train[:,i])
x_train[:,7] = labelencoder_x.fit_transform(x_train[:,7])

[17] labelencoder_y= LabelEncoder()
     y_train= labelencoder_y.fit_transform(y_train)

[19] for i in range(0, 5):
       x_test[:,i] = labelencoder_x.fit_transform(x_test[:,i])
     x_test[:,7] = labelencoder_x.fit_transform(x_test[:,7])

[20] labelencoder_y= LabelEncoder()
     y_test= labelencoder_y.fit_transform(y_test)
```

**Figure 18:** Changing all the categorical variables to numbers

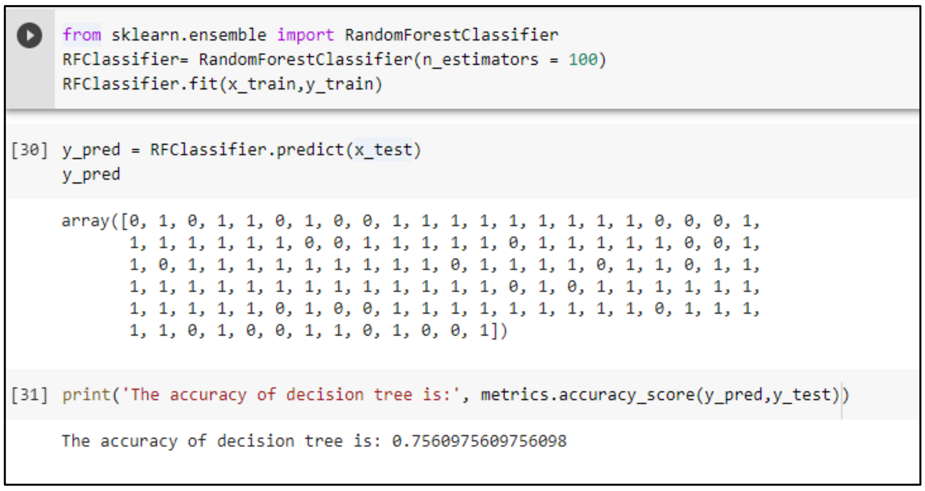
All the categorical variables are now turned into numbers (Fig. 18). So, now we will scale our data as it improves our prediction.Now we will test different classification models for accuracy. First model we will test is the Decision tree (Fig. 19).



```
from sklearn.tree import DecisionTreeClassifier
DTClassifier= DecisionTreeClassifier(criterion='entropy', random_state=0)
DTClassifier.fit(x_train,y_train)

[23] y_pred= DTClassifier.predict(x_test)
     y_pred

     array([0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1,
            1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1,
            1, 1, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 1, 1, 0, 0, 1, 0, 1, 1,
            1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1,
            1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1,
            1, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1])

[25] from sklearn import metrics
     print('The accuracy of decision tree is:', metrics.accuracy_score(y_pred,y_test))

     The accuracy of decision tree is: 0.6991869918699187
```

**Figure 19**: Decision tree accuracy test

Now let’s see the accuracy for Random Forest (Fig. 20).



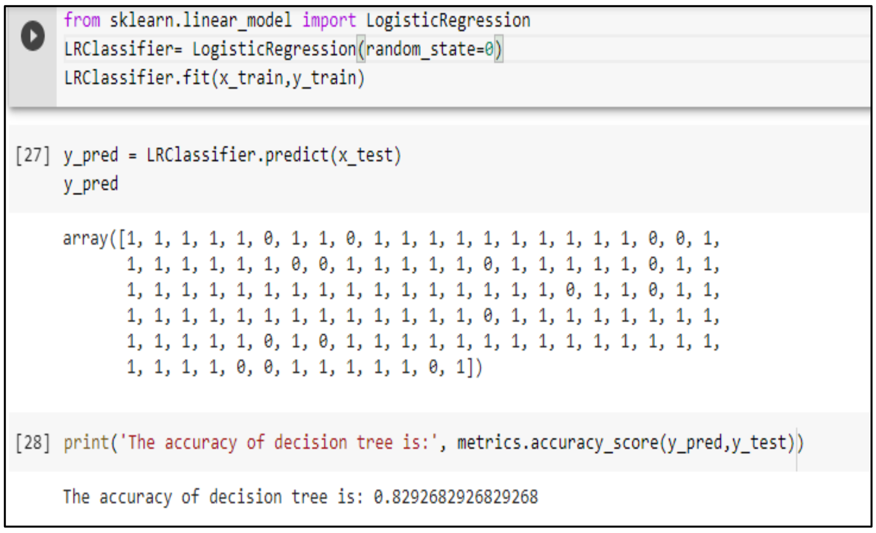
```
from sklearn.ensemble import RandomForestClassifier
RFClassifier= RandomForestClassifier(n_estimators = 100)
RFClassifier.fit(x_train,y_train)
```

```
[30] y_pred = RFClassifier.predict(x_test)
     y_pred

     array([0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1,
            1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1,
            1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1,
            1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1,
            1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1,
            1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 1])
```

```
[31] print('The accuracy of decision tree is:', metrics.accuracy_score(y_pred,y_test))

     The accuracy of decision tree is: 0.7560975609756098
```

**Figure 20:** Random Forest accuracy test

This model has a greater accuracy than the decision tree. Now let’s test for Logistic Regression model (Fig. 21).



```
from sklearn.linear_model import LogisticRegression
LRClassifier= LogisticRegression(random_state=0)
LRClassifier.fit(x_train,y_train)
```

```
[27] y_pred = LRClassifier.predict(x_test)
     y_pred

     array([1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1,
            1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1,
            1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1,
            1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1,
            1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
            1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1])
```

```
[28] print('The accuracy of decision tree is:', metrics.accuracy_score(y_pred,y_test))

     The accuracy of decision tree is: 0.8292682926829268
```

**Figure 21**: Logistic Regression accuracy test

Accuracy for logistic regression is higher than the DT algorithm and RF algorithm. It is a good model to be used for loan prediction.

## 4. Conclusion

All in all, in this paper the three models that is Logistic Regression, Decision Tree and Random Forest were applied so as to build loan prediction models that will forecast the loan approval status of applicants as Yes or No. After training and testing all the three models with train dataset and test dataset we had the accuracy values on the basis of which we came to a conclusion that Logistic Regression algorithm will be accurate in predicting loan as it had a high accuracy value. Using this algorithm we will be able to predict the right applicants for the loan approval with proper accuracy.

## 5. References

[1] G. Arutjothi and Dr. C. Senthamaria. “Prediction of Loan Status in Commercial Bank using Machine Learning Classifier”, International Conference on Intelligent Sustainable Systems(ICISS 2017). doi:10.1109/iss1.2017.8389442.

[2] Aboobyda Jafar Hamid and Tarig Mohammed Ahmed. “DEVELOPING PREDICTION MODEL OF LOAN RISK IN BANKS USING DATA MINING”, Machine Learning and Applications: An International Journal (MLAIJ) Vol.3, No.1, March 2016.

[3] Kumar Arun, Garg Ishan, Kaur Sanmeet. “Loan Approval Prediction based on Machine Learning Approach", National Conference on Recent Trends in Computer Science and Information Technology (NCERT CSIT-2016).

[4] Louis W.Glorfeld and Bill C.Hardgrave. “An improved method for developing neural networks: The case of evaluating commercial loan creditworthiness”, Computers & Operations Research, Volume 23, Issue 10, October 1996.

[5] Andy Liaw and Matthew Wiener. “Classification and Regression by randomForest”, ISSN 1609-3631, Vol. 2/3,December2002.

[6] Stephan Dreiseitl and Lucila Ohno-Machado. “Logistic regression and artificial neural network classification models: a methodology review”, Journal of Biomedical Informatics, Volume 35, Issues 5–6, October 2002.

[7] T. Choudhury, G. Dangi, T. P. Singh, A. Chauhan and A. Aggarwal, "An Efficient Way to Detect Credit Card Fraud Using Machine Learning Methodologies," 2018 Second International Conference on Green Computing and Internet of Things (ICGCIoT), 2018, pp. 591-597, doi: 10.1109/ICGCIoT.2018.8753077.

[8] S. Taneja, D. Garg, M. V. Tarun Kumar and T. Choudhury, "The Machine Predicted Market," 2018 International Conference on Computational Techniques, Electronics and Mechanical Systems (CTEMS), 2018, pp. 256-260, doi: 10.1109/CTEMS.2018.8769306.

[9] Sharma, H.K., Choudhury, T., Toe, T.T. (2022). Machine Learning Based Predictive Analytics: A Use Case in Insurance Sector. In: Jeyanthi, P.M., Choudhury, T., Hack-Polay, D., Singh, T.P., Abujar, S. (eds) Decision Intelligence Analytics and the Implementation of Strategic Business Management. EAI/Springer Innovations in Communication and Computing. Springer, Cham. https://doi.org/10.1007/978-3-030-82763-2_14.