PROJECT REPORT

LOAN STATUS PREDICTION

Submitted by

HARSHINI S

IMARTICUS LEARNING

ABSTRACT

The cost of assets is increasing day by day and the capital required to purchase an entire asset is very high. So purchasing it out of your savings is not possible. The easiest way to get the required funds is to apply for a loan. But taking a loan is a very time consuming process. The application has to go through a lot of stages and it's still not necessary that it will be approved. To decrease the approval time and to decrease the risk associated with the loan many loan prediction models were introduced. The aim of this project was to compare the various Loan Prediction Models and show which is the best one with the least amount of error and could be used by banks in real world to predict if the loan should be approved or not taking the risk factor in mind. After comparing and analyzing the models, it was found that the prediction model based on LogisticRegression proved to be the most accurate and fitting of them all. This can be useful in reducing the time and manpower required to approve loans and filter out the perfect candidates for providing loans.

Dataset Link: https://github.com/Harshinisasikumar12/CAPSTONE-DATASET-

Source Code: https://github.com/Harshinisasikumar12/LAON-STATUS-PREDICTION-CAPSTONE-PROJECT

ACKNOWLEDGEMENT

I am using this opportunity to express my gratitude to everyone who supported me throughout the course of my capstone project. I am thankful for their aspiring guidance, invaluably constructive criticism and friendly advice during the project work. I am sincerely grateful to them for sharing their truthful and illuminating views on a number of issues related to the project.

Further, I have fortunate to have Mr.PRASAD as my mentor. He has readily shared his immense knowledge in data analytics and guide me in a manner that the outcome resulted in enhancing my data skills.

I wish to thank all the faculties, as this project utilized knowledge gained from every course that formed the DSP program.

I certify that the work done by me for conceptualizing and completing this project is original and authentic.

Date: July 10, 2022 Name: HARSHINI S

CERTIFICATE OF COMPLETION

I hereby certify that the project titled “Loan Status Prediction” was undertaken and completed the project (10th July, 2022).

Mentor : Mr. Prasad

Date : 10th July,2022

Place : Karur

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

TITLE & OBJECTIVE OF THE STUDY:

- ✓ A Prediction Model uses data mining, statistics and probability to forecast an outcome. Every model has some variables known as predictors that are likely to influence future results. The data that was collected from various resources then a statistical model is made. It can use a simple linear equation or a sophisticated neural network mapped using a complex software. The Prediction Model helps the banks by minimizing the risk associated with the loan approval system and helps the applicant by decreasing the time taken in the process.
- ✓ The main objective of the Project is to compare the Loan Prediction Models made implemented using various algorithms and choose the best one out of them that can shorten the loan approval time and decrease the risk associated with it. It is done by predicting if the loan can be given to that person on the basis of various parameters like credit score, income, age, marital status, gender, etc. The prediction model not only helps the applicant but also helps the bank by minimizing the risk and reducing the number of defaulters.

Figure:1.1Loan Status

1.2 BUSINESS OR ENTERPRISE UNDER STUDY

Loan Approval Prediction based on Machine Learning Approach" Author- Kumar Arun, Garg Ishan, Kaur Sanmeet Year- 2018The 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. Exploring the Machine Learning Algorithm for Prediction the Loan Sanctioning Process" Author- E. Chandra Blessie, R. Rekha - Year- 2019 Extending credits to corporates and individuals for the smooth functioning of growing economies like India is inevitable.

As increasing number of customers apply for loans in the banks and non- banking financial companies (NBFC), it is really challenging for banks and NBFCs with limited capital to device a standard resolution and safe procedure to lend money to its borrowers for their financial needs. In addition, in recent times NBFC inventories have suffered a significant downfall in terms of the stock price. It has contributed to a contagion that has also spread to other financial stocks, adversely affecting the benchmark in recent times. In this paper, an attempt is made to condense the risk involved in selecting the suitable person who could repay the loan on time thereby keeping the bank's nonperforming assets (NPA)on the hold. This is achieved by feeding the past records of the customer who acquired loans from the bank into a trained machine learning model which could yield an accurate result. The prime focus of the paper is to determine whether or not it will be safe to allocate the loan to a particular person.

This is achieved by feeding the past records of the customer who acquired loans from the bank into a trained machine learning model which could yield an accurate result. The prime focus of the paper is to determine whether or not it will be safe to allocate the loan to a particular person.

CHAPTER 2

DATA COLLECTION & PREPARATION

The dataset collected for foretelling loan failure clients is foretold into Training set and testing set. Generally 8020 proportion is applied to dissociate the training set and testing set. The data model which was created using Logistic Regression and SVM is applied on the training set and hung on the test take fineness, Test set forecasting is done. Following are the attributes.

2.1 LIST OF VARIABLES IN DATA

VARIABLE	DESCRIPTION
LOAN_ID	Unique loan id
GENDER	Male / female
MARRIED	Applicant married(Y/N)
DEPENDENTS	Number of dependents
EDUCATION	(Graduate/under graduate)
SELF_EMPLOYED	Self employed(Y/N)

APPLICANTINCOME	Applicant income
COAPPLICATIONINCOME	Co applicationincome
LOANAMOUNT	Loan amount inthousands
LOAN_AMOUNT_TERM	Term of loan inmonths

CHAPTER 3

EXPLORATORY DATA ANALYSIS

I have done all preprocessing steps and also the visualization techniques for better understanding and to check assumptions with the help of statistical summary and graphical representations of the health measures of the people to give the better outcomes.

3.1 DATA PREPROCESSING

Data preprocessing is the process of transforming raw data into an understandable format. It is also an important step in data mining as we cannot work with raw data.

In this model, I have done many preprocessing steps. The goal is to explore,investigate and learn as opposed to confirming statistical hypothesis.Some of the processes includes descriptive statistical summary of the given data, checking null values, checking the duplicates and correlation to check how the variables are related to each other.

```
In [10]: df.describe()
```

Out[10]:

	Dependents	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	614.000000	614.000000	614.000000	614.000000
mean	1.454397	5403.459283	1623.491726	147.239414	339.016287	0.817590
std	1.089939	6109.041673	2925.531126	87.060346	68.106311	0.386497
min	0.000000	150.000000	0.000000	9.000000	12.000000	0.000000
25%	0.000000	2877.500000	0.000000	100.000000	360.000000	1.000000
50%	1.000000	3812.500000	1211.500000	128.000000	360.000000	1.000000
75%	2.000000	5795.000000	2297.250000	167.750000	360.000000	1.000000
max	3.000000	81000.000000	41667.000000	700.000000	480.000000	1.000000

Figure:3.1.1 Descriptive summary

CHECKING NULL VALUES

```
In [12]: df.isnull().sum()

Out[12]: 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
```

Figure:3.1.2 Checking null values

CHECKING DUPLICATES

```
In [13]: df.duplicated()

Out[13]: 0      False
         1      False
         2      False
         3      False
         4      False
                ...
         609    False
         610    False
         611    False
         612    False
         613    False
         Length: 614, dtype: bool

In [14]: df.duplicated().sum()

Out[14]: 0
```

Figure:3.1.3 Checking duplicates

```
In [36]: corr=df.corr()
         corr
```

Out[36]:

	Dependents	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
Dependents	1.000000	0.041854	-0.010457	0.096552	-0.036338	0.015074
ApplicantIncome	0.041854	1.000000	-0.116577	0.539539	-0.034860	-0.005619
CoapplicantIncome	-0.010457	-0.116577	1.000000	0.184765	-0.064834	-0.052931
LoanAmount	0.096552	0.539539	0.184765	1.000000	0.034228	-0.033073
Loan_Amount_Term	-0.036338	-0.034860	-0.064834	0.034228	1.000000	0.061033
Credit_History	0.015074	-0.005619	-0.052931	-0.033073	0.061033	1.000000

Figure:3.1.4 Displaying correlation values

3.2.DATA VISUALIZATION

Data Visualization is the practice of translating information into a visual context such as a map, graph, etc…., These visualizations used to figure out how data is used in a particular machine learning model it helps in analyzing it.

In this model,I have done many visualization methods.These visual displays of information communicate complex data relationships and data-driven insights in a way that is easy to understand.

Some of the visualization includes Count plot for category representation, Bar chart for comparing different variables and analyzing patterns over long period of time.

System approve the loan if documents are cleared and reject the loan if documents are not cleared Report is delivered to the applicant according to their status.

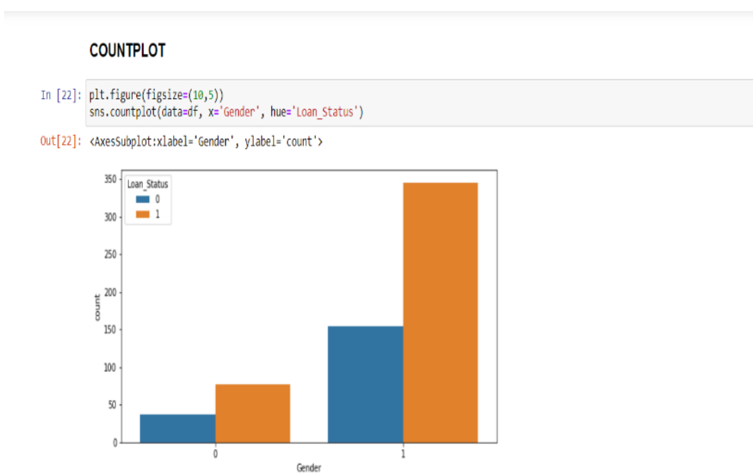


Figure:3.2.1 Count plot

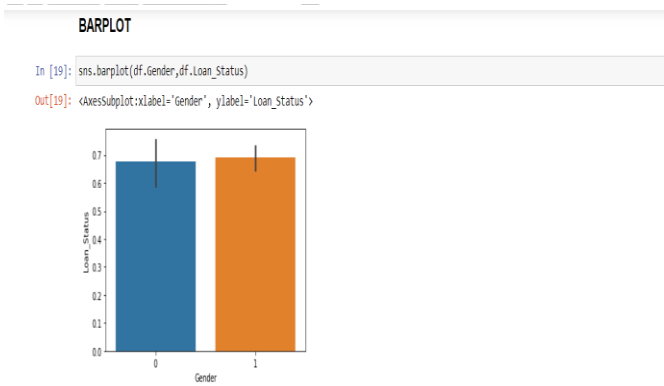


Figure:3.2.2 Bar plot

Figure:3.2.3 Visualizing Using histogram

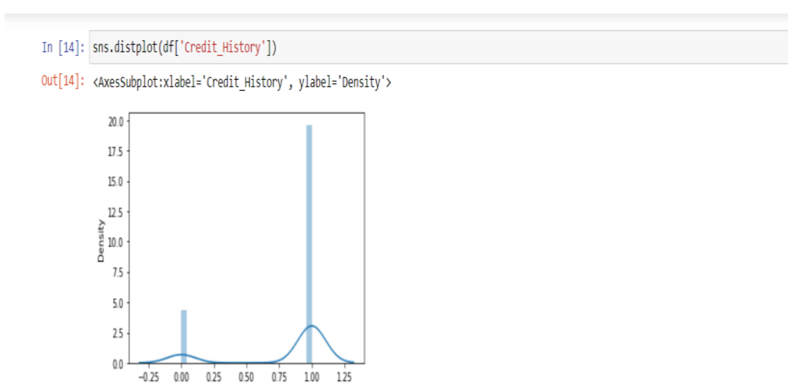


Figure:3.2.4 Distribution plot

CHAPTER 4

TRAIN & TEST VALIDATION SPLIT

The goal of training is to answer a question or make a prediction correctly as often as possible. Now we should train the model on the training dataset and make sooth sayings for the test dataset. We can divide our train dataset into two tract train and testimony. We can train the model on this training part and using that make sooth sayings for the testimony part. In this way, we can validate our sooth sayings as we've the true sooth sayings for the testimony part (which we don't have for the test dataset).

In [17]:
```
X=df.drop(['Loan_Status'],axis=1)
Y=df['Loan_Status']
```

TESTING AND TRAINING THE MODEL

In [69]:
```
from sklearn.model_selection import train_test_split
X_train, X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.1,stratify=Y,random_state=None)
```

In [70]:
```
print(X.shape, X_train.shape, X_test.shape)
```

```
(614, 13) (552, 13) (62, 13)
```

Figure:4.1 Train & Test Validation

CHAPTER 5

FITTING MODELS TO DATA

5.1 LOGISTIC REGRESSION

The project is to predict whether the person is eligible for getting loan or not. So for this problem, I use Classification method called Logistic Regression. Logistic regression is an example of supervised learning. It is used to calculate or predict the probability of a binary (yes/no) event occurring.

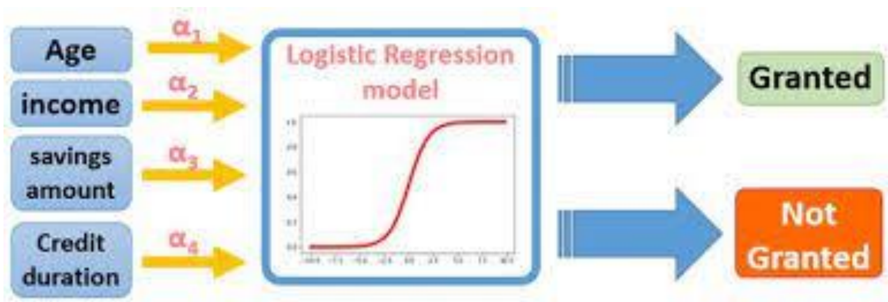


Figure:5.1 Logistic Regression

I have used this Logistic Regression algorithm in my model. In addition to that, I have checked a Accuracy Score.

```
In [91]: from sklearn.linear_model import LogisticRegression
         data=LogisticRegression()
         data.fit(X_train,Y_train)

Out[91]: ▸ LogisticRegression

In [ ]:  proba=data.predict(X_test)
         data.predict_proba(X_test)

In [33]: y_pred=data.predict(X_test)
         y_pred_train=data.predict(X_train)
         print(y_pred)

         [1 1 1 1 1 0 1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 0 1 0 1 1 1 0 0 1 1 1 1 1 1 0 0
          1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 0 1 1]

In [34]: from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
         a=accuracy_score(Y_test,proba)
         print(a)

         0.8225806451612904
```

Figure:5.1.2 Fitting data to Logistic Regression

I have tested my model with validation data which have give the accuracy of nearly 82%.

5.2 SUPPORT VECTOR MACHINE

In this approach, each data item is plotted in a n-dimensional space, where n represents the number of features with each feature represented in a corresponding co- ordinates. A hyper plane is determined to distinguish the classes (possibly two) based on their features.

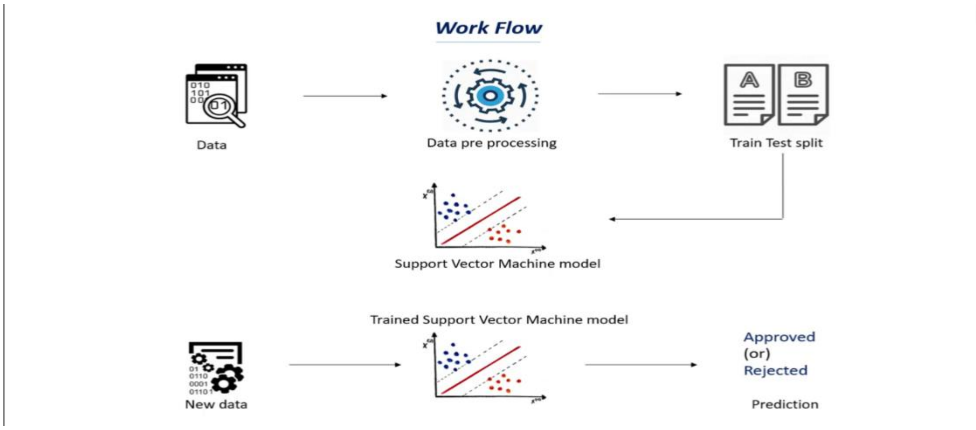


Figure:5.2.1 Support Vector Machine

USING SUPPORT VECTOR MACHINE TO PREDICT ACCURACY

In [29]:
```
from sklearn import svm
classifier = svm.SVC(kernel='linear')
```

In [ ]:
```
classifier.fit(X_train,Y_train)
```

In [ ]:
```
y_pred=classifier.predict(X_test)
```

In [35]:
```
from sklearn.metrics import accuracy_score
accuracy_score(Y_test,y_pred)
```

Out[35]:
```
0.8064516129032258
```

In [36]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(Y_test,y_pred)
```

Out[36]:
```
array([[12,  7],
       [ 4, 39]], dtype=int64)
```

In [37]:
```
from sklearn.metrics import classification_report
print(classification_report(Y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.75      0.63      0.69        19
           1       0.85      0.91      0.88        43

    accuracy                           0.82        62
   macro avg       0.80      0.77      0.78        62
weighted avg       0.82      0.82      0.82        62
```

Figure:5.2.2 Fitting data to Support Vector Machine

5.3 HYPERPARAMATER TUNING

Hyperparameter tuning consists of finding a set of optimal hyperparameter values for a learning algorithm while applying this optimized algorithm to any data set. That combination of hyperparameters maximizes the model's performance, minimizing a predefined loss function to produce better results with fewer errors for improving the accuracy I used hyperparameter tuning for logistic regression.

HYPERPARAMETER TUNING

```
This is used to imporve the accuracy performance
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
# Creating the hyperparameter grid
c_space = np.logspace(-5, 8, 15)
param_grid = {'C': c_space}
# Instantiating logistic regression classifier
logreg = LogisticRegression()
# Instantiating the GridSearchCV object
logreg_cv = GridSearchCV(logreg, param_grid, cv = 5)
logreg_cv.fit(X, Y)
print("Best score is {}".format(logreg_cv.best_score_))
```

```
Best score is 0.9739999999999999
```

Figure5.3.1 HyperParameter Tuning

By using hyperparameter tuning in logistic regression it gives the accuracy of 97%

CHAPTER 6

PREDICTING A NEW DATA

I have taken a loan status for a person who is eligible for loan are not and tested with my logistic regression model have predict the loan status accurately.

PREDICTION FOR A MODEL

In [34]:
```
Y=data.predict([[26,1,0,3,1,0,7920,8346,540,470,1,3,8000]])
if(Y==1):
    print("LOAN APPROVED FOR A PERSON")
else:
    print("LOAN REJECTED FOR A PERSON")
```

```
LOAN APPROVED FOR A PERSON
```

Figure:6.1 Prediction for a model

CHAPTER 7

KEY FINDINGS

MODEL NAME	ACCURACY SCORE
Logistic Regression	0.82
Support Vector Machine	0.80
Hyperparameter Tuning For Logistic Regression	0.97

From above table we analyze the logistic regression is best model for loan status prediction.

LOAN STATUS PREDICTION

DESCRIPTION OF THE PROJECT

It is a classification problem , given information about the application we have to predict whether the they'll be to pay the loan or not. The 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

In [8]:

```
import pandas as pd
dataset=pd.read_csv("C:/Users/Harshini/Downloads/Loan Prediction.csv")
df=pd.DataFrame(dataset)
df
```

Out[8]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplicantl
0	LP001002	Male	No	2	Graduate	No	5849	
1	LP001003	Male	Yes	3	Graduate	No	4583	
2	LP001005	Male	Yes	2	Graduate	Yes	3000	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplicantl
4	LP001008	Male	No	2	Graduate	No	6000	
...	...	...	...	...	...	...	...	
609	LP002978	Female	No	2	Graduate	No	2900	
610	LP002979	Male	Yes	0	Graduate	No	4106	
611	LP002983	Male	Yes	0	Graduate	No	8072	
612	LP002984	Male	Yes	2	Graduate	No	7583	
613	LP002990	Female	No	0	Graduate	Yes	4583	

614 rows × 13 columns

In [9]:

```
from warnings import filterwarnings
filterwarnings("ignore")
```

In [10]:

```
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             614 non-null    object
 2   Married            614 non-null    object
 3   Dependents         614 non-null    int64
 4   Education          614 non-null    object
 5   Self_Employed      614 non-null    object
 6   ApplicantIncome    614 non-null    int64
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         614 non-null    int64
 9   Loan_Amount_Term   614 non-null    int64
 10  Credit_History     614 non-null    int64
 11  Property_Area      614 non-null    object
 12  Loan_Status        614 non-null    object
dtypes: float64(1), int64(5), object(7)
memory usage: 62.5+ KB
```

In [11]:

```
df.columns
```

Out[11]:

```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Status'],
      dtype='object')
```

In [12]:

```
df.dtypes
```

Out[12]:

```
Loan_ID              object
Gender               object
```

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

In [13]:

```
df.describe()
```

Out[13]:

	Dependents	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_Hist
count	614.000000	614.000000	614.000000	614.000000	614.000000	614.000
mean	1.454397	5403.459283	1623.491726	147.239414	339.016287	0.817
std	1.089939	6109.041673	2925.531126	87.060346	68.106311	0.386
min	0.000000	150.000000	0.000000	9.000000	12.000000	0.000
25%	0.000000	2877.500000	0.000000	100.000000	360.000000	1.000
50%	1.000000	3812.500000	1211.500000	128.000000	360.000000	1.000
75%	2.000000	5795.000000	2297.250000	167.750000	360.000000	1.000
max	3.000000	81000.000000	41667.000000	700.000000	480.000000	1.000

In [14]:

```
df.shape
```

Out[14]:

```
(614, 13)
```

CHECKING NULL VALUES

In [15]:

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

Out[15]:

```
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
```

CHECKING DUPLICATES

In [16]:
```
df.duplicated()
```

Out[16]:
```
0      False
1      False
2      False
3      False
4      False
       ...
609    False
610    False
611    False
612    False
613    False
Length: 614, dtype: bool
```

In [17]:
```
df.duplicated().sum()
```

Out[17]:
```
0
```

PRINTING THE FIRST FIVE ROWS

In [18]:
```
df.head()
```

Out[18]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantInc
0	LP001002	Male	No	2	Graduate	No	5849	1:
1	LP001003	Male	Yes	3	Graduate	No	4583	1!
2	LP001005	Male	Yes	2	Graduate	Yes	3000	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2:
4	LP001008	Male	No	2	Graduate	No	6000	

PRINTING THE LAST FIVE ROWS

In [19]:
```
df.tail()
```

Out[19]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	Coapplicantl
609	LP002978	Female	No	2	Graduate	No	2900	
610	LP002979	Male	Yes	0	Graduate	No	4106	
611	LP002983	Male	Yes	0	Graduate	No	8072	
612	LP002984	Male	Yes	2	Graduate	No	7583	
613	LP002990	Female	No	0	Graduate	Yes	4583	

CONVERTING CATEGORICAL VALUES INTO NUMERICAL VALUES

In [20]:

```
df.replace({'Married':{'No':0,'Yes':1},'Gender':{'Male':1,'Female':0},'Self_Employed':{
                      'Property_Area':{'Rural':0,'Semiurban':1,'Urban':2},'Education':{
```

In [21]:

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()

for i in df:
    if df[i].dtype=='object':
        df[i]=le.fit_transform(df[i])
```

In [19]:

```
df.head()
```

Out[19]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantInco
0	0	1	0	2	1	0	5849	137
1	1	1	1	3	1	0	4583	150
2	2	1	1	2	1	1	3000	
3	3	1	1	0	0	0	2583	235
4	4	1	0	2	1	0	6000	

CREATION OF NEW ATTRIBURES

In [22]:

```
df['Total_Income'] = df['ApplicantIncome'] + df['CoapplicantIncome']
df
```

Out[22]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIn
0	0	1	0	2	1	0	5849	
1	1	1	1	3	1	0	4583	
2	2	1	1	2	1	1	3000	
3	3	1	1	0	0	0	2583	
4	4	1	0	2	1	0	6000	
...	...	...	...	...	...	...	...	
609	609	0	0	2	1	0	2900	
610	610	1	1	0	1	0	4106	
611	611	1	1	0	1	0	8072	
612	612	1	1	2	1	0	7583	

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIn
613	613	0	0	0	1	1	4583	

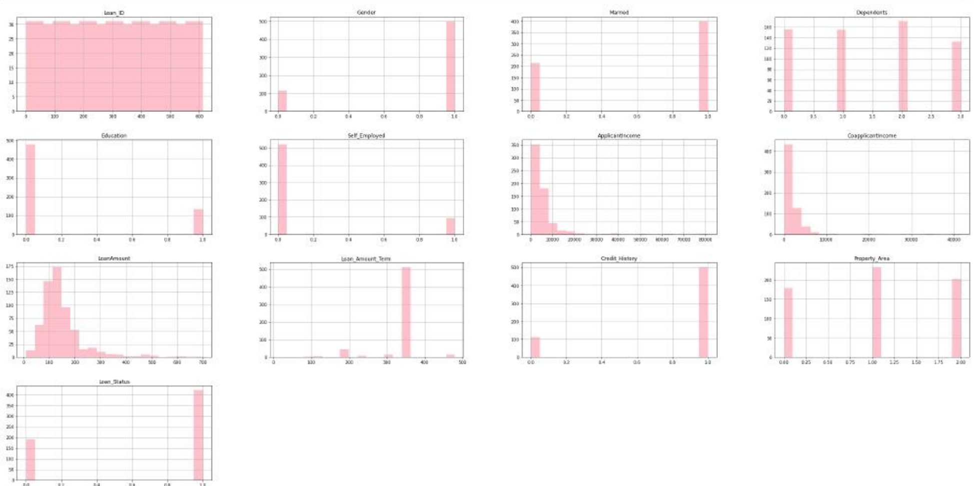
614 rows × 14 columns

EXPLORATORY DATA ANALYSIS

In [ ]:
```
Exploratory Data Analysis is an approach to analyze the data using visual techniques.
It is used to discover trends, patterns, or to check assumptions with the help of stati
```

In [13]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

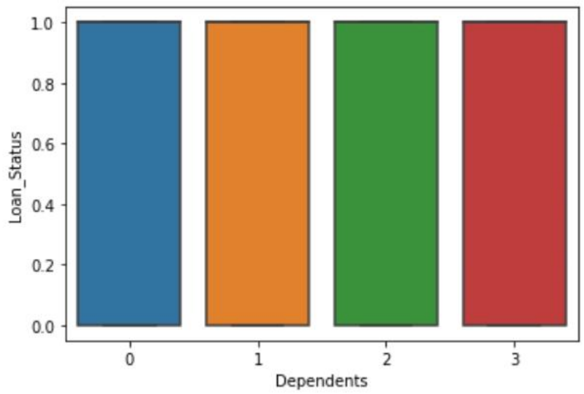
In [15]:
```
df.hist(bins=20,figsize=(40,20),color='pink')
plt.show()
```



BOXPLOT

In [31]:
```
sns.boxplot(data=df,x="Dependents",y="Loan_Status")
```

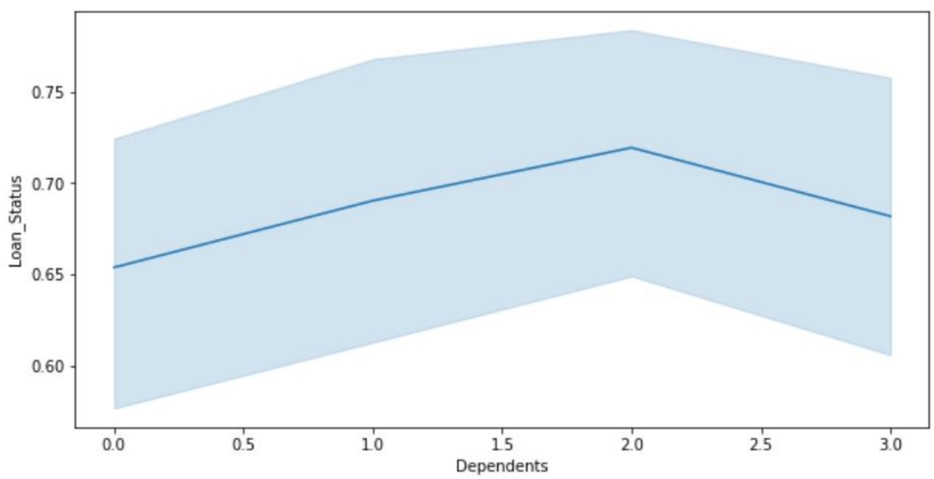
Out[31]: <AxesSubplot:xlabel='Dependents', ylabel='Loan_Status'>



LINEPLOT

In [24]:
```
plt.figure(figsize=(10,5))
sns.lineplot(data=df, x='Dependents', y='Loan_Status')
```

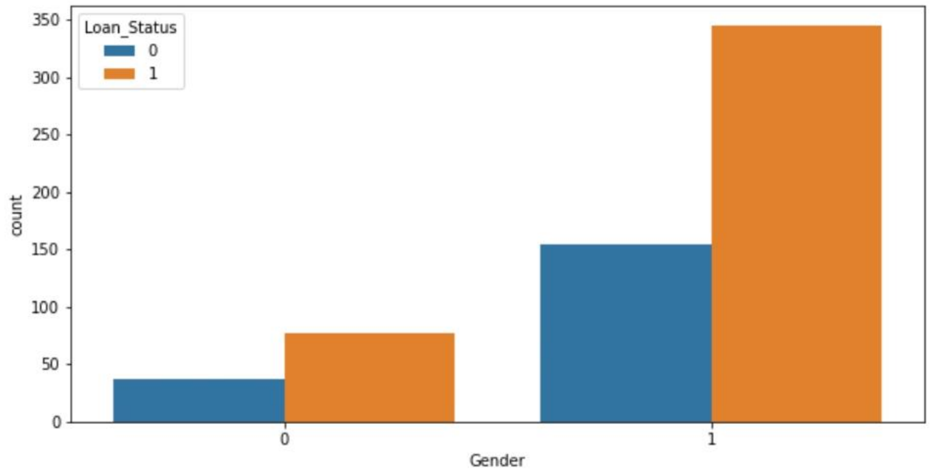
Out[24]: <AxesSubplot:xlabel='Dependents', ylabel='Loan_Status'>



COUNTPLOT

In [22]:
```
plt.figure(figsize=(10,5))
sns.countplot(data=df, x='Gender', hue='Loan_Status')
```

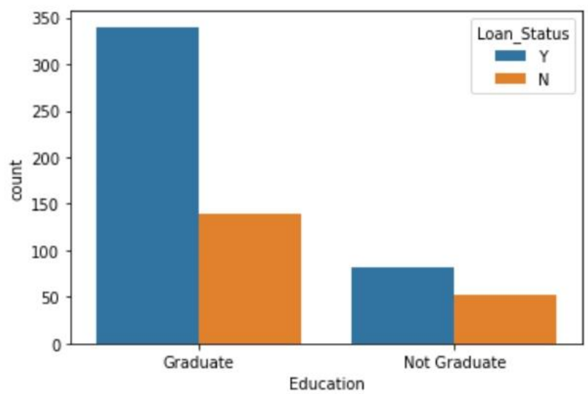
Out[22]: <AxesSubplot:xlabel='Gender', ylabel='count'>



In [15]:

```
sns.countplot(x='Education',hue='Loan_Status',data=df)
```

Out[15]: <AxesSubplot:xlabel='Education', ylabel='count'>

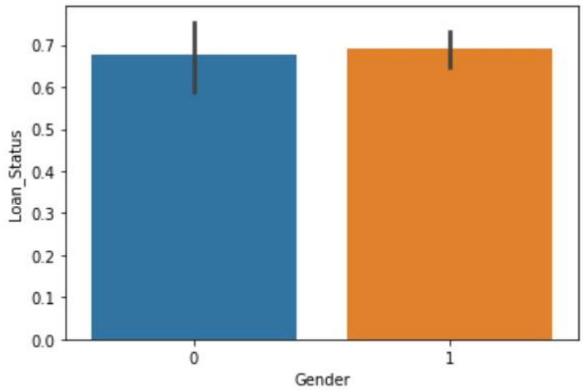


BARPLOT

In [19]:

```
sns.barplot(df.Gender,df.Loan_Status)
```

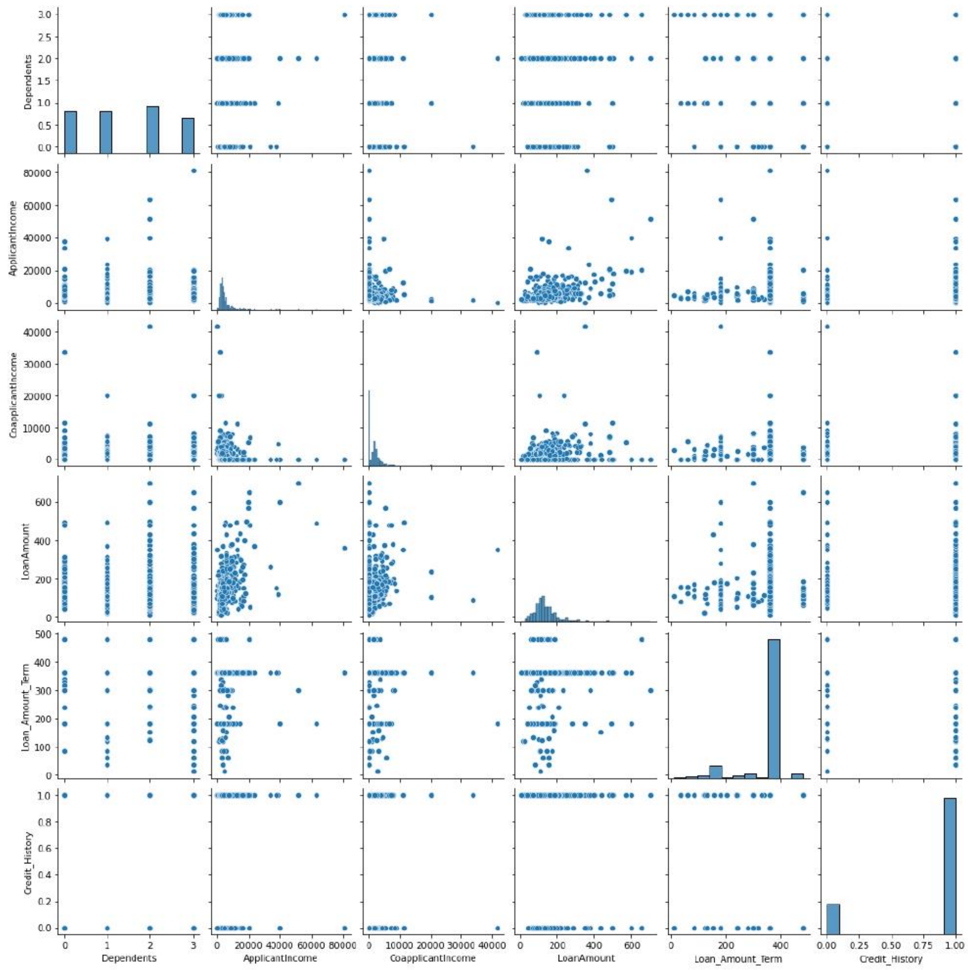
Out[19]: <AxesSubplot:xlabel='Gender', ylabel='Loan_Status'>



VISUALIZATION USING PAIRPLOT

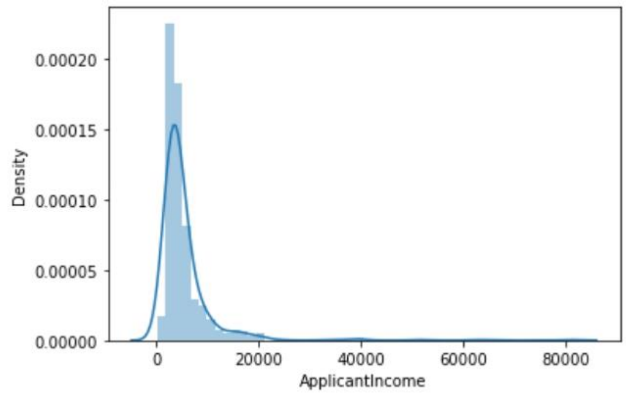
In [42]:

```
import seaborn as sns
sns.pairplot(df)
plt.show()
```



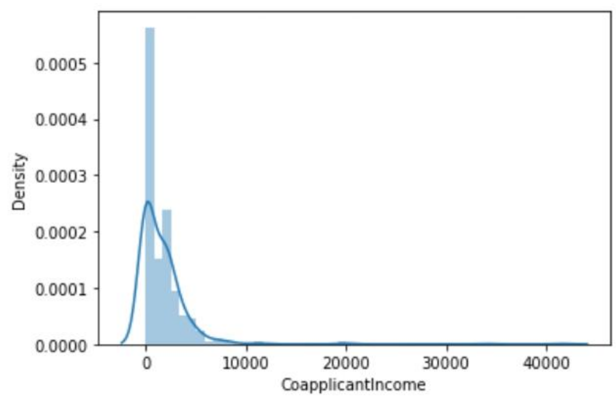
In [12]:
```
import seaborn as sns
sns.distplot(df["ApplicantIncome"])
```

Out[12]: <AxesSubplot:xlabel='ApplicantIncome', ylabel='Density'>



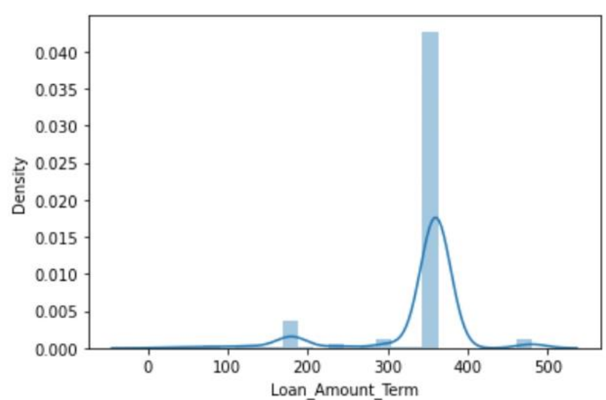
In [13]:
```
sns.distplot(df["CoapplicantIncome"])
```

Out[13]: <AxesSubplot:xlabel='CoapplicantIncome', ylabel='Density'>



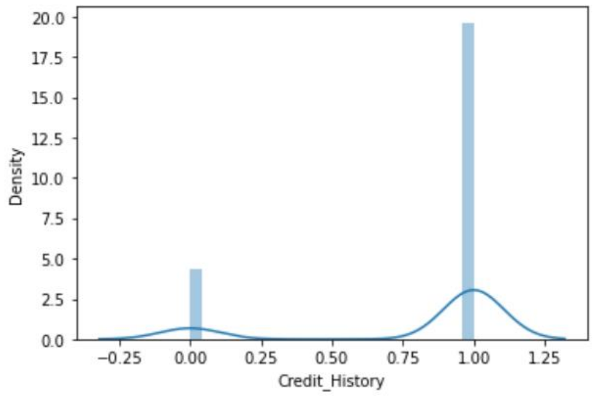
In [15]:
```
sns.distplot(df['Loan_Amount_Term'])
```

Out[15]: <AxesSubplot:xlabel='Loan_Amount_Term', ylabel='Density'>



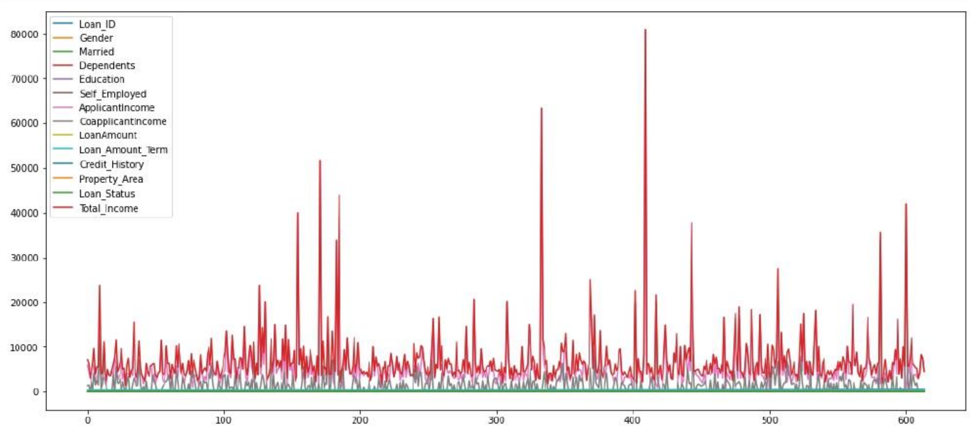
In [14]:
```
sns.distplot(df['Credit_History'])
```

Out[14]: <AxesSubplot:xlabel='Credit_History', ylabel='Density'>



In [44]:
```
import matplotlib.pyplot as plt
df.plot(figsize=(18, 8))

plt.show()
```

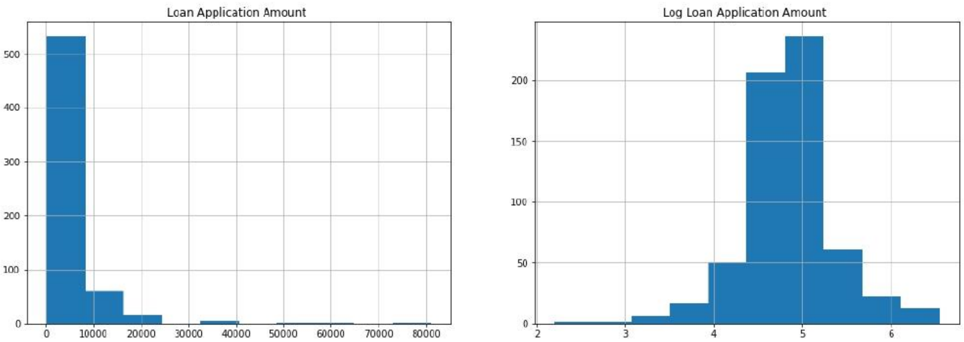


In [46]:
```
import numpy as np
plt.figure(figsize=(18, 6))
plt.subplot(1, 2, 1)


df['ApplicantIncome'].hist(bins=10)
plt.title("Loan Application Amount ")

plt.subplot(1, 2, 2)
plt.grid()
plt.hist(np.log(df['LoanAmount']))
plt.title("Log Loan Application Amount ")
```

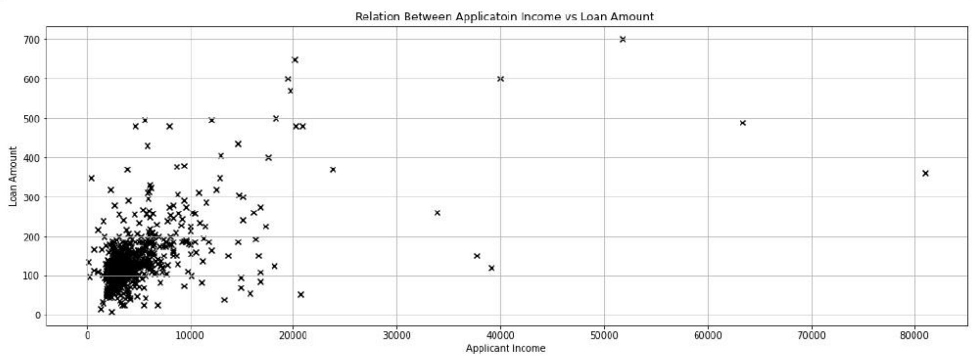
```
plt.show()
```



In [48]:

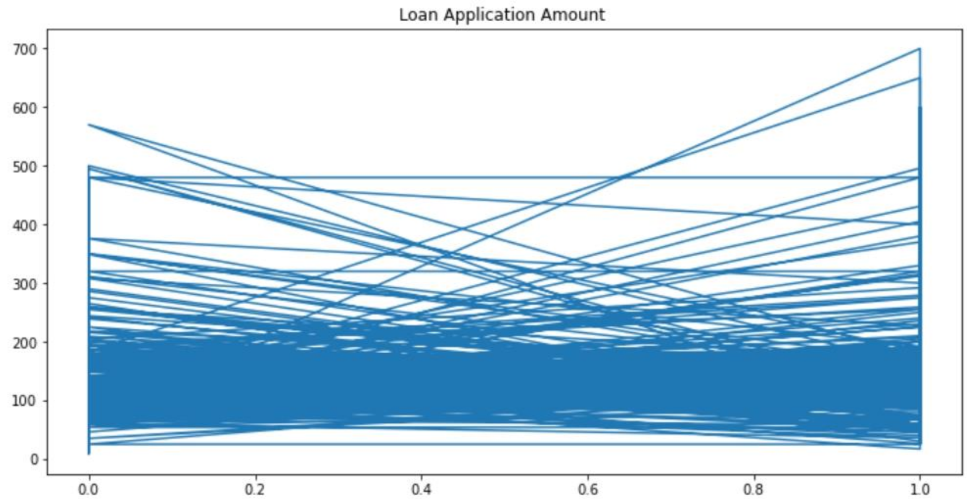
```
plt.figure(figsize=(18, 6))
plt.title("Relation Between Applicatoin Income vs Loan Amount ")

plt.grid()
plt.scatter(df['ApplicantIncome'] , df['LoanAmount'], c='k', marker='x')
plt.xlabel("Applicant Income")
plt.ylabel("Loan Amount")
plt.show()
```



In [49]:

```
plt.figure(figsize=(12, 6))
plt.plot(df['Loan_Status'], df['LoanAmount'])
plt.title("Loan Application Amount ")
plt.show()
```



CORRELATION

In [36]:

```
corr=df.corr()
corr
```

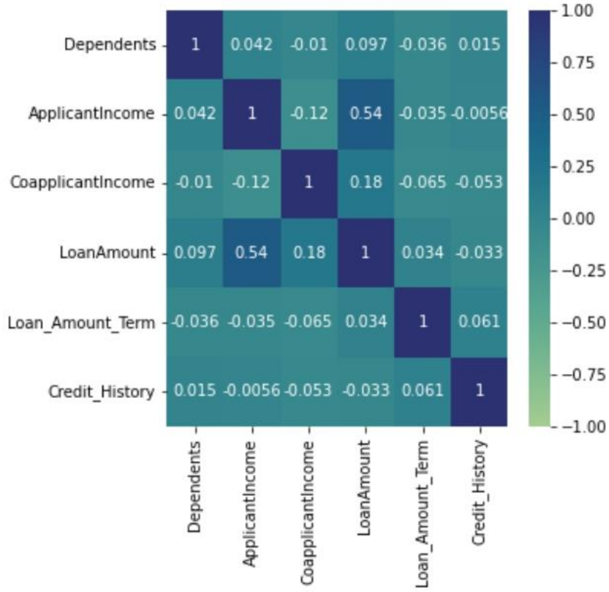
Out[36]:

	Dependents	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term
Dependents	1.000000	0.041854	-0.010457	0.096552	-0.036338
ApplicantIncome	0.041854	1.000000	-0.116577	0.539539	-0.034860
CoapplicantIncome	-0.010457	-0.116577	1.000000	0.184765	-0.064834
LoanAmount	0.096552	0.539539	0.184765	1.000000	0.034228
Loan_Amount_Term	-0.036338	-0.034860	-0.064834	0.034228	1.000000
Credit_History	0.015074	-0.005619	-0.052931	-0.033073	0.061033

In [37]:

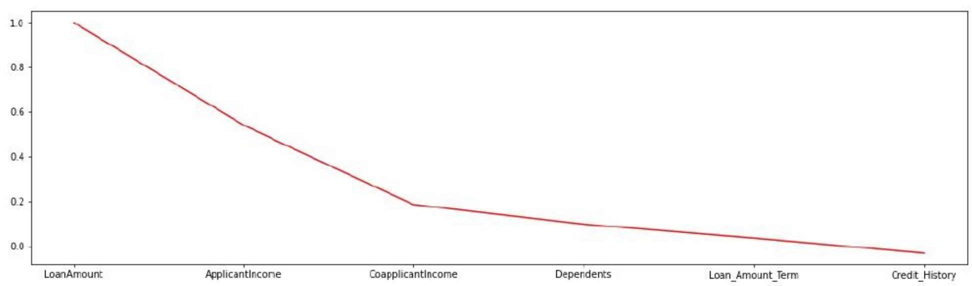
```
import matplotlib.pyplot as plt
import seaborn as sns
plt.figure(figsize=(5,5))
sns.heatmap(corr, annot=True, vmin=-1, cmap='crest')
```

Out[37]: <AxesSubplot:>



In [38]:
```
plt.figure(figsize=(18,5))
corr['LoanAmount'].sort_values(ascending=False).plot(color='r')
```

Out[38]: <AxesSubplot:>



In [37]:
```
X=df.drop(['Loan_Status'],axis=1)
Y=df['Loan_Status']
```

TESTING AND TRAINING THE MODEL

In [38]:
```
from sklearn.model_selection import train_test_split
X_train, X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.1,stratify=Y,random_s
```

In [39]:
```
print(X.shape, X_train.shape, X_test.shape)
```

```
(614, 13) (552, 13) (62, 13)
```

USING SUPPORT VECTOR MACHINE TO PREDICT ACCURACY

In [41]:
```
from sklearn import svm
classifier = svm.SVC(kernel='linear')
```

In [42]:
```
classifier.fit(X_train,Y_train)
```

Out[42]:
```
▾ SVC
SVC(kernel='linear')
```

In [ ]:
```
y_pred=classifier.predict(X_test)
```

In [35]:
```
from sklearn.metrics import accuracy_score
accuracy_score(Y_test,y_pred)
```

Out[35]:
```
0.8064516129032258
```

In [36]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(Y_test,y_pred)
```

Out[36]:
```
array([[12,  7],
       [ 4, 39]], dtype=int64)
```

In [37]:
```
from sklearn.metrics import classification_report
print(classification_report(Y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.75      0.63      0.69        19
           1       0.85      0.91      0.88        43

    accuracy                           0.82        62
   macro avg       0.80      0.77      0.78        62
weighted avg       0.82      0.82      0.82        62
```

TO PREDICT ACCURACY USING LOGISTIC REGRESSION

In [26]:
```
from sklearn.linear_model import LogisticRegression
data=LogisticRegression()
data.fit(X_train,Y_train)
```

Out[26]:
```
▾ LogisticRegression
LogisticRegression()
```

In [27]:
```
X_test
```

Out[27]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIn
268	268	0	0	0	1	0	3418	
505	505	1	1	2	1	0	3510	
337	337	1	1	0	1	1	2500	
446	446	1	1	1	0	0	4652	
425	425	1	1	0	1	0	2666	
...	...	...	...	...	...	...	...	
444	444	1	1	3	1	0	7333	
316	316	1	1	2	1	0	3717	
280	280	1	0	2	0	1	4053	
227	227	1	1	3	1	0	6250	
111	111	0	1	0	1	0	2929	

62 rows × 13 columns

In [28]:

```
proba=data.predict(X_test)
data.predict_proba(X_test)
```

Out[28]:

```
array([[0.46843628, 0.53156372],
       [0.2893922 , 0.7106078 ],
       [0.38717736, 0.61282264],
       [0.32875043, 0.67124957],
       [0.35986956, 0.64013044],
       [0.56975702, 0.43024298],
       [0.28250558, 0.71749442],
       [0.18803763, 0.81196237],
       [0.32249557, 0.67750443],
       [0.15155065, 0.84844935],
       [0.33811454, 0.66188546],
       [0.06996138, 0.93003862],
       [0.59581113, 0.40418887],
       [0.21772754, 0.78227246],
       [0.67062533, 0.32937467],
       [0.31794423, 0.68205577],
       [0.08181562, 0.91818438],
       [0.75848872, 0.24151128],
       [0.17145431, 0.82854569],
       [0.2228814 , 0.7771186 ],
       [0.18676544, 0.81323456],
       [0.66574953, 0.33425047],
       [0.13924764, 0.86075236],
       [0.62784922, 0.37215078],
       [0.21170985, 0.78829015],
       [0.19263459, 0.80736541],
       [0.08428642, 0.91571358],
       [0.69231222, 0.30768778],
       [0.68767236, 0.31232764],
       [0.22318624, 0.77681376],
```

```
       [0.27168633, 0.72831367],
       [0.33406836, 0.66593164],
       [0.18551595, 0.81448405],
       [0.09624095, 0.90375905],
       [0.26139329, 0.73860671],
       [0.6929212 , 0.3070788 ],
       [0.53917537, 0.46082463],
       [0.38675037, 0.61324963],
       [0.22590227, 0.77409773],
       [0.60168381, 0.39831619],
       [0.37006659, 0.62993341],
       [0.21198689, 0.78801311],
       [0.45590244, 0.54409756],
       [0.27195329, 0.72804671],
       [0.32206884, 0.67793116],
       [0.15822467, 0.84177533],
       [0.37106438, 0.62893562],
       [0.75402933, 0.24597067],
       [0.18981544, 0.81018456],
       [0.35692653, 0.64307347],
       [0.19719534, 0.80280466],
       [0.1227878 , 0.8772122 ],
       [0.34690754, 0.65309246],
       [0.4853158 , 0.5146842 ],
       [0.06027557, 0.93972443],
       [0.25112841, 0.74887159],
       [0.16552985, 0.83447015],
       [0.54682336, 0.45317664],
       [0.16618954, 0.83381046],
       [0.60903631, 0.39096369],
       [0.16973948, 0.83026052],
       [0.33072668, 0.66927332]])
```

In [62]:

```
y_pred=data.predict(X_test)
y_pred_train=data.predict(X_train)
print(y_pred)
```

```
[1 1 1 1 1 0 1 1 1 1 1 1 0 1 0 1 1 0 1 1 1 0 1 0 1 1 1 0 0 1 1 1 1 1 1 0 0
 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 0 1 1]
```

In [63]:

```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
a=accuracy_score(Y_test,proba)
print(a)
```

```
0.8225806451612904
```

In [66]:

```
import seaborn as sns
from sklearn.metrics import confusion_matrix
confusion_matrix(Y_test,proba)
```

Out[66]:

```
array([[11,  8],
       [ 3, 40]], dtype=int64)
```

In [30]:

```
print(classification_report(Y_test,proba))
```

```
              precision    recall  f1-score   support
```

CHAPTER 8

CONCLUSION

The task of this machine learning project is to train the model for accepting loan or rejecting loan. Now there are 2 models where in we can train the model and test to predict whether other applicants could get loan or not. So here, it can be concluded with confidence that the Logistic Regression model is extremely efficient and gives a better result when compared to other models. It works correctly and fulfills all requirements of bankers. This system properly and accurately calculate the result. It predicts the loan is approve or reject to loan applicant or customer very accurately. From a proper analysis of positive points and constraints on the member, it can be safely concluded that the product is a considerably productive member. This use is working duly and meeting to all Banker requisites. This member can be freely plugged in numerous other systems. There have been mathematics cases of computer glitches, violations in content and most important weight of features is fixed in automated prophecy system, so in the near future the so – called software could be made more secure, trustworthy and dynamic weight conformation. In near future this module of prophecy can be integrated with the module of automated processing system. The system is trained on old training dataset in future software can be made resembling that new testing date should also take part in training data after some fix time.

CHAPTER 9

REFERENCE

[1] Kumar Arun, Garg Ishan, Kaur Sanmeet, May-Jun. 2016. Loan Approval Prediction based on Machine Learning Approach, IOSR Journal of Computer Engineering (IOSR-JCE)

[2] Wei Li, Shuai Ding, Yi Chen, and Shanlin Yang, Heterogeneous Ensemble for Default Prediction of Peer-to-Peer Lending in China, Key Laboratory of Process Optimization and Intelligent DecisionMaking, Ministry of Education, Hefei University of Technology, Hefei 2009, China

[3] Short-term prediction of Mortgage default using ensembled machine learning models, Jesse C. Sealand on july 20, 2018.

[4] Clustering Loan Applicants based on Risk Percentage using K-Means Clustering Techniques, Dr. K. Kavitha, International Journal of Advanced Research in Computer Science and Software Engineering.

[5] Kumar Arun, Garg Ishan, Kaur Sanmeet, ―Loan Approval Prediction based on Machine Learning Approach‖, IOSR Journal of Computer Engineering (IOSR-JCE), Vol. 18, Issue 3, pp. 79-81, Ver. I (May-Jun. 2016).

[6] Hassan, Amira Kamil Ibrahim and Ajith Abraham. "Modeling consumer loan default prediction using ensemble neural networks, 2013 International Conference On Computing, Electrical And Electronic Engineering (ICCEEE). IEEE, 2013