Name:-P.S.V.RAMARAJU Mail:-rrpinnamraju@gmail.com

In [1]:
```
import warnings
warnings.filterwarnings("ignore")
```

(1)-Importing all the necessary libraries

In [2]:
```
# (1)-Importing all the necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
import nltk
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
```

(2)-Importing the dataset provided

In [3]:
```
#Reading the dataset provided with extension of ".csv"
dt=pd.read_csv(r"C:\Users\RamaRaju\Desktop\loan-predictionUC.csv")
```

In [4]:
```
#printing the data from the dataset
dt
```

Out[4]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0
...	...	...	...	...	...	...	...	...	...	...
609	LP002978	Female	No	0	Graduate	No	2900	0.0	71.0	360.0
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180.0
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0	253.0	360.0
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	187.0	360.0
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0

614 rows × 13 columns

(3)-Understanding the data

In [5]:
```
dt.describe()           #describes the data
```

Out[5]:

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

In [6]:
```
dt.info()               #it gives the data information
```

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

In [7]:
```
dt.head(10)          #prints the top 10 values
```

Out[7]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	C
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	
5	LP001011	Male	Yes	2	Graduate	Yes	5417	4196.0	267.0	360.0	
6	LP001013	Male	Yes	0	Not Graduate	No	2333	1516.0	95.0	360.0	
7	LP001014	Male	Yes	3+	Graduate	No	3036	2504.0	158.0	360.0	
8	LP001018	Male	Yes	2	Graduate	No	4006	1526.0	168.0	360.0	
9	LP001020	Male	Yes	1	Graduate	No	12841	10968.0	349.0	360.0	

In [8]:
```
dt.tail(10)                    #prints the bottom 10 values
```

Out[8]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term
604	LP002959	Female	Yes	1	Graduate	No	12000	0.0	496.0	360.0
605	LP002960	Male	Yes	0	Not Graduate	No	2400	3800.0	NaN	180.0
606	LP002961	Male	Yes	1	Graduate	No	3400	2500.0	173.0	360.0
607	LP002964	Male	Yes	2	Not Graduate	No	3987	1411.0	157.0	360.0
608	LP002974	Male	Yes	0	Graduate	No	3232	1950.0	108.0	360.0
609	LP002978	Female	No	0	Graduate	No	2900	0.0	71.0	360.0
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180.0
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0	253.0	360.0
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	187.0	360.0
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0

In [9]:
```
dt['Loan_ID'].unique()            #prints the every single values from "Loan_ID"
```

Out[9]:
```
array(['LP001002', 'LP001003', 'LP001005', 'LP001006', 'LP001008',
       'LP001011', 'LP001013', 'LP001014', 'LP001018', 'LP001020',
       'LP001024', 'LP001027', 'LP001028', 'LP001029', 'LP001030',
       'LP001032', 'LP001034', 'LP001036', 'LP001038', 'LP001041',
       'LP001043', 'LP001046', 'LP001047', 'LP001050', 'LP001052',
       'LP001066', 'LP001068', 'LP001073', 'LP001086', 'LP001087',
       'LP001091', 'LP001095', 'LP001097', 'LP001098', 'LP001100',
       'LP001106', 'LP001109', 'LP001112', 'LP001114', 'LP001116',
       'LP001119', 'LP001120', 'LP001123', 'LP001131', 'LP001136',
       'LP001137', 'LP001138', 'LP001144', 'LP001146', 'LP001151',
       'LP001155', 'LP001157', 'LP001164', 'LP001179', 'LP001186',
       'LP001194', 'LP001195', 'LP001197', 'LP001198', 'LP001199',
       'LP001205', 'LP001206', 'LP001207', 'LP001213', 'LP001222',
       'LP001225', 'LP001228', 'LP001233', 'LP001238', 'LP001241',
       'LP001243', 'LP001245', 'LP001248', 'LP001250', 'LP001253',
       'LP001255', 'LP001256', 'LP001259', 'LP001263', 'LP001264',
       'LP001265', 'LP001266', 'LP001267', 'LP001273', 'LP001275',
       'LP001279', 'LP001280', 'LP001282', 'LP001289', 'LP001310',
       'LP001316', 'LP001318', 'LP001319', 'LP001322', 'LP001325',
```

In [11]:
```
dt['Gender'].unique()            #Provides the every single value in "Gender"
```

Out[11]:
```
array(['Male', 'Female', nan], dtype=object)
```

In [12]:
```
dt.shape                          #Shows the Shape of the data
```

Out[12]:
```
(614, 13)
```

In [15]:
```
list(dt)                    #list the titles of the data
```

Out[15]:
```
['Loan_ID',
 'Gender',
 'Married',
 'Dependents',
 'Education',
 'Self_Employed',
 'ApplicantIncome',
 'CoapplicantIncome',
 'LoanAmount',
 'Loan_Amount_Term',
 'Credit_History',
 'Property_Area',
 'Loan_Status']
```

In [16]:
```
dt1=dt[["LoanAmount"]]          #Reads the Values in LoanAmount and assign to dt1
dt1                             #prints the values in dt1
```

Out[16]:

	LoanAmount
0	NaN
1	128.0
2	66.0
3	120.0
4	141.0
...	...
609	71.0
610	40.0
611	253.0
612	187.0
613	133.0

614 rows × 1 columns

In [17]:
```
dt1.max()                    #prints the maximum value in dt1
```

Out[17]:
```
LoanAmount    700.0
dtype: float64
```

In [18]:
```
dt1.min()                    #prints the minimum value in dt1
```

Out[18]:
```
LoanAmount    9.0
dtype: float64
```

(4)-Dealing with the missing values .

In [19]:
```
#finding null values
dt2=dt.isna().sum()
dt2
```

Out[19]:
```
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
```

In [20]:
```
#filling null values with some values.
dt.fillna(35,inplace=True)
dt
```

Out[20]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term
0	LP001002	Male	No	0	Graduate	No	5849	0.0	35.0	360.0
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0
...	...	...	...	...	...	...	...	...	...	...
609	LP002978	Female	No	0	Graduate	No	2900	0.0	71.0	360.0
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180.0
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0	253.0	360.0
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	187.0	360.0
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0

614 rows × 13 columns

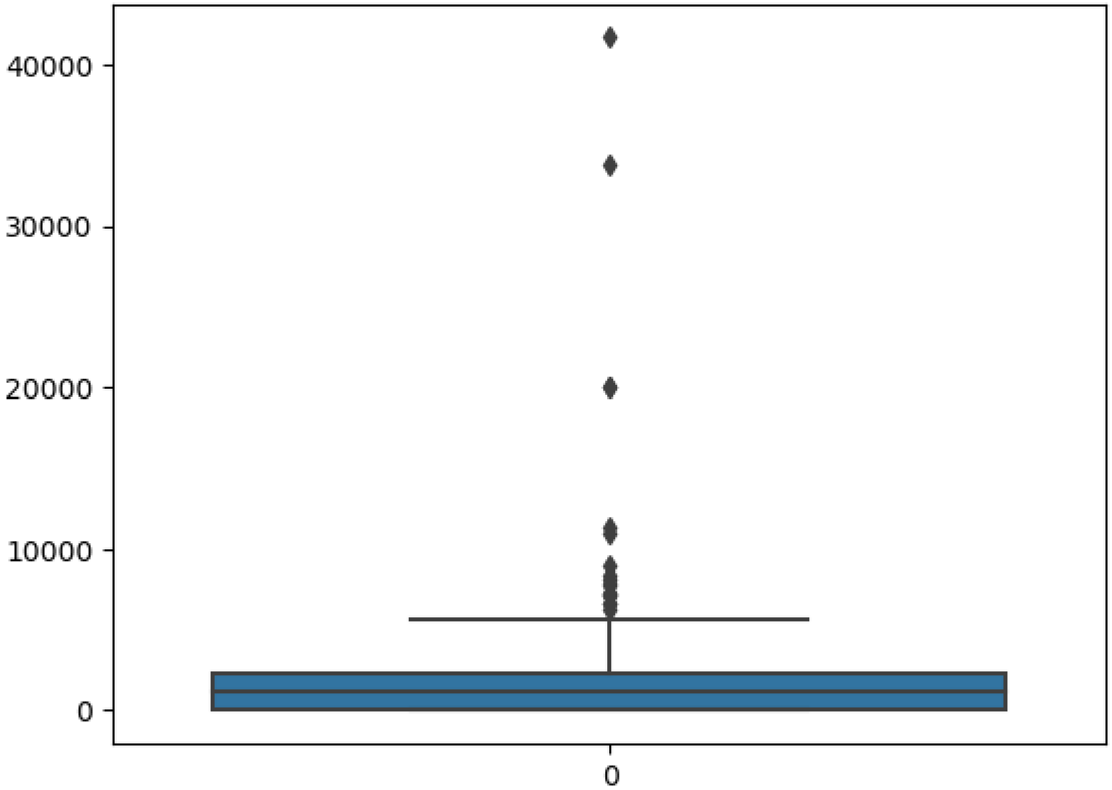
In [21]:
```
#Now all the null values are filled.so,we get nullvalues=0.
dt3=dt.isna().sum()
dt3
```

Out[21]:
```
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
```

(5)-Visualization

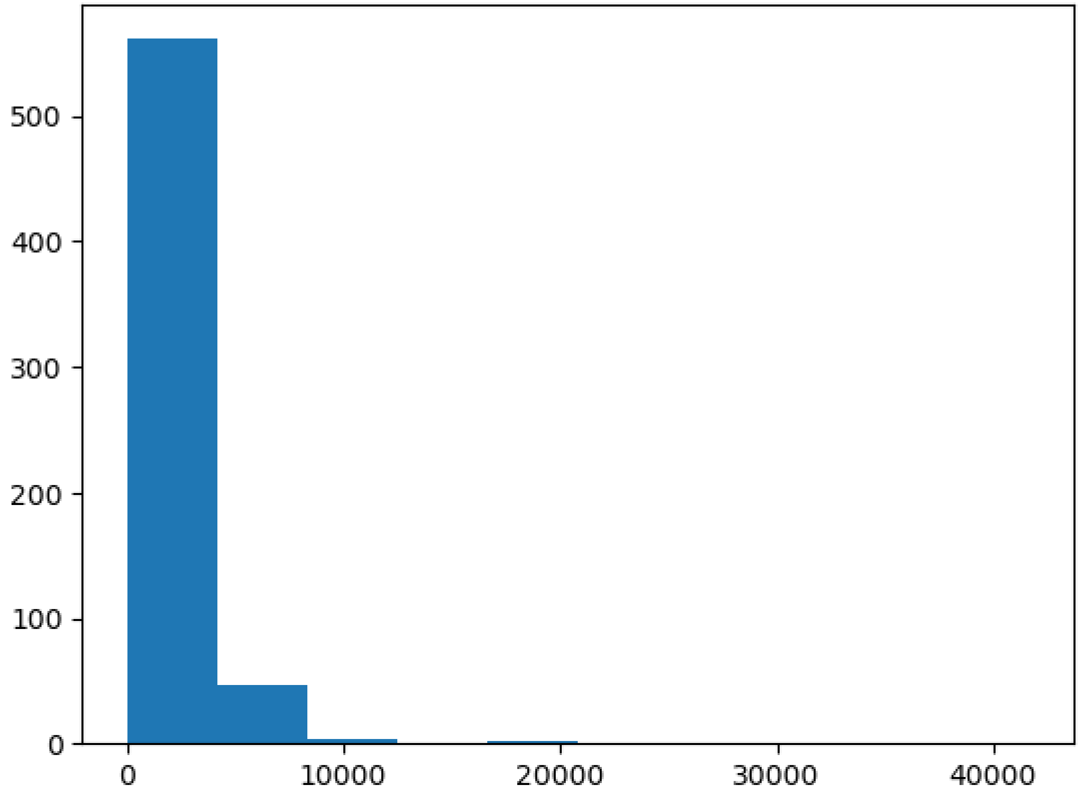
In [22]:
```
#boxplot from seaborn
sns.boxplot(dt.CoapplicantIncome)
```

Out[22]: <Axes: >



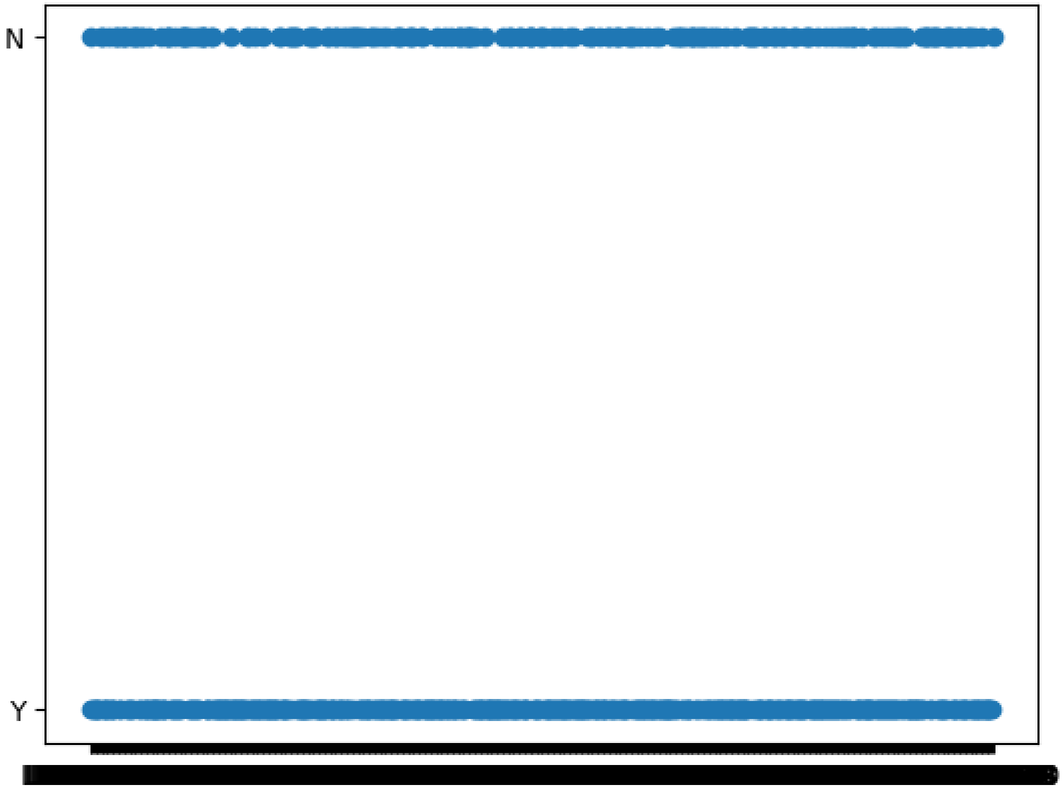
In [23]:
```
#plotting using matplot library.
plt.hist(dt['CoapplicantIncome'])
```

Out[23]:
```
(array([561.,  46.,   3.,   0.,   2.,   0.,   0.,   0.,   1.,   1.]),
 array([    0. ,  4166.7,  8333.4, 12500.1, 16666.8, 20833.5, 25000.2,
        29166.9, 33333.6, 37500.3, 41667. ]),
 <BarContainer object of 10 artists>)
```



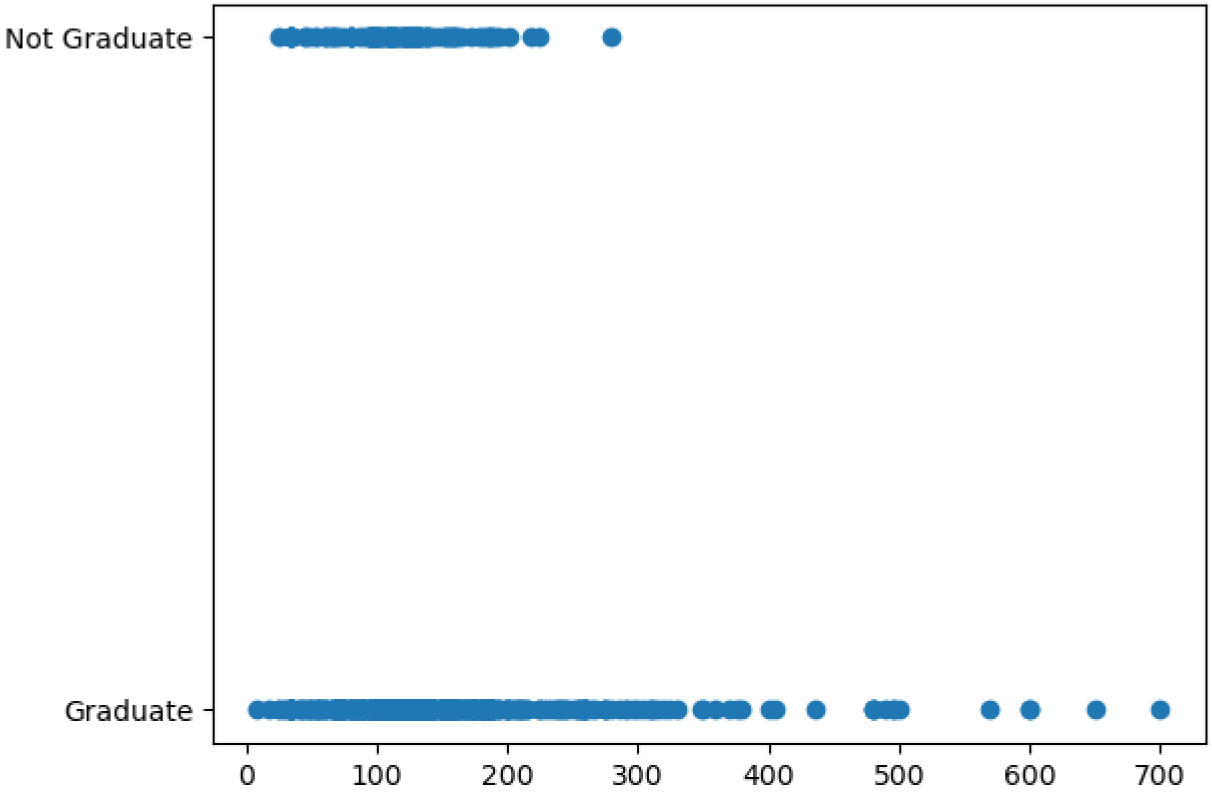
In [28]:
```
#scatterplot:- Used to know the releationship b/n two variables in dots(In 2-D).
#scatterplot b/n Loan_ID and Loan_Status
import matplotlib.pyplot as plt
%matplotlib inline
plt.scatter(dt["Loan_ID"],dt["Loan_Status"])
```

Out[28]:
```
<matplotlib.collections.PathCollection at 0x2ce48f86290>
```



In [29]:
```
#Scatterplot b/n LoanAmount and Education
import matplotlib.pyplot as plt
%matplotlib inline
plt.scatter(dt["LoanAmount"],dt["Education"])
```

Out[29]:
```
<matplotlib.collections.PathCollection at 0x2ce498bd510>
```



In [30]:

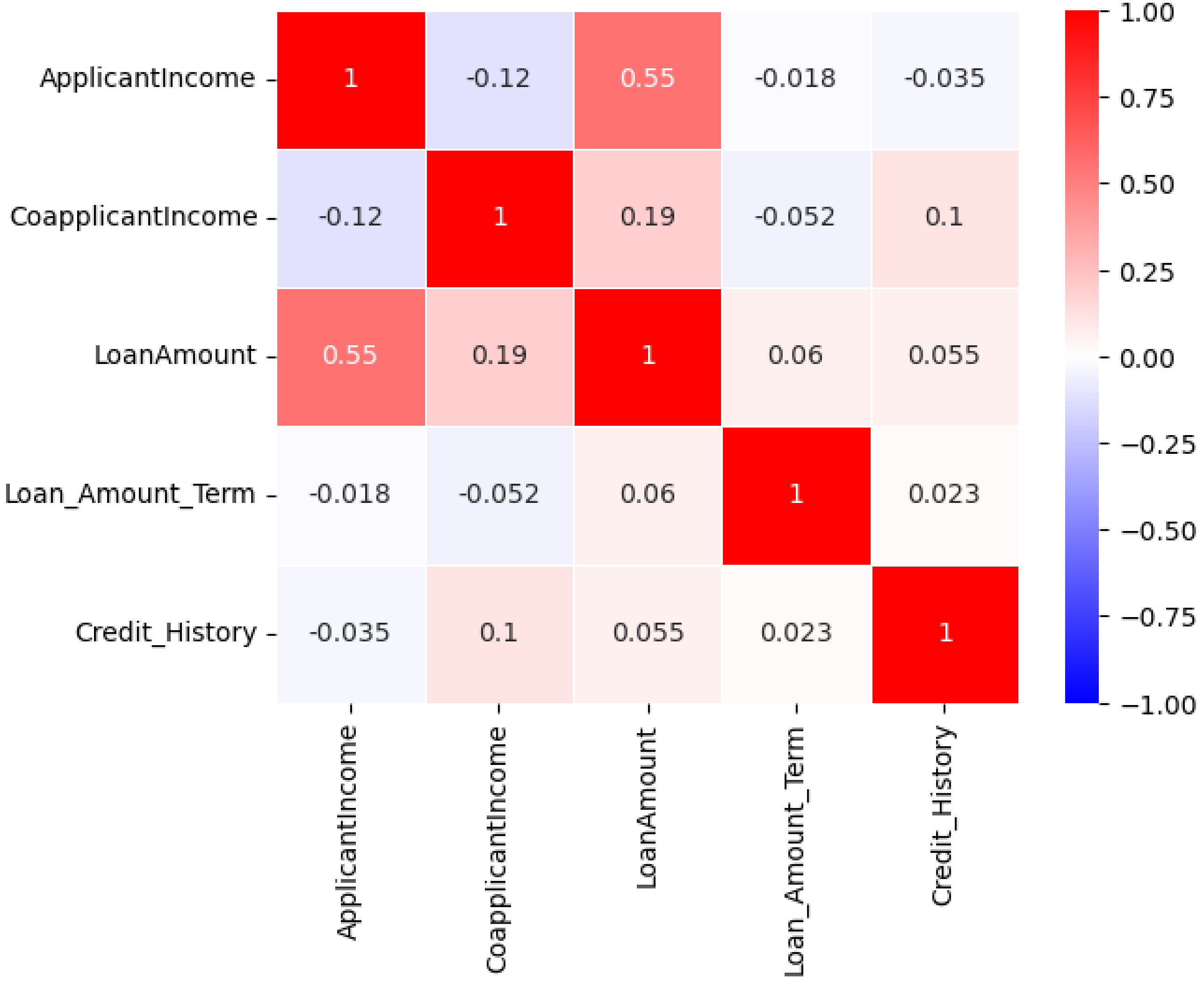
```
#Correlation:- Used To know the releation b/n the variables in the dataset
cor=dt.corr()
cor
```

Out[30]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
ApplicantIncome	1.000000	-0.116605	0.549339	-0.018374	-0.035168
CoapplicantIncome	-0.116605	1.000000	0.191450	-0.051942	0.104162
LoanAmount	0.549339	0.191450	1.000000	0.059771	0.055400
Loan_Amount_Term	-0.018374	-0.051942	0.059771	1.000000	0.022653
Credit_History	-0.035168	0.104162	0.055400	0.022653	1.000000

In [31]:
```
#HeatMap:- Used to Visualize the strength of Correlation among the variables.
sns.heatmap(cor,vmax=1,vmin=-1,annot=True,linewidths=.5,cmap='bwr')
```

Out[31]: <Axes: >



(6)- Dividing the dataset into training and test datasets

In [32]:
```
#Grouping the data based on the rows and columns.
dt.groupby('Loan_Status').count()
```

Out[32]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amoun
Loan_Status										
N	192	192	192	192	192	192	192	192	192	
Y	422	422	422	422	422	422	422	422	422	

In [33]:
```
#Assigning the "X" and "y" variables.
y=dt['Loan_Status']
X=dt.drop('Loan_Status',axis=1)
```

In [34]:
```
y                  #printing the values in "y"
```

Out[34]:
```
0      Y
1      N
2      Y
3      Y
4      Y
      ..
609    Y
610    Y
611    Y
612    Y
613    N
Name: Loan_Status, Length: 614, dtype: object
```

In [35]:
```
X                    #printing the values in "X"
```

Out[35]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term
0	LP001002	Male	No	0	Graduate	No	5849	0.0	35.0	360.0
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0
...	...	...	...	...	...	...	...	...	...	...
609	LP002978	Female	No	0	Graduate	No	2900	0.0	71.0	360.0
610	LP002979	Male	Yes	3+	Graduate	No	4106	0.0	40.0	180.0
611	LP002983	Male	Yes	1	Graduate	No	8072	240.0	253.0	360.0
612	LP002984	Male	Yes	2	Graduate	No	7583	0.0	187.0	360.0
613	LP002990	Female	No	0	Graduate	Yes	4583	0.0	133.0	360.0

614 rows × 12 columns

In [36]:
```
#Drop:- Dropping or Removing of the entire column from the data.
dt4=dt.drop(["Gender"],axis=1)
dt4
```

Out[36]:

	Loan_ID	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_H
0	LP001002	No	0	Graduate	No	5849	0.0	35.0	360.0	
1	LP001003	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	
2	LP001005	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	
3	LP001006	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	
4	LP001008	No	0	Graduate	No	6000	0.0	141.0	360.0	
...	...	...	...	...	...	...	...	...	...	
609	LP002978	No	0	Graduate	No	2900	0.0	71.0	360.0	
610	LP002979	Yes	3+	Graduate	No	4106	0.0	40.0	180.0	
611	LP002983	Yes	1	Graduate	No	8072	240.0	253.0	360.0	
612	LP002984	Yes	2	Graduate	No	7583	0.0	187.0	360.0	
613	LP002990	No	0	Graduate	Yes	4583	0.0	133.0	360.0	

614 rows × 12 columns

In [37]:
```
#get_dummies is used to convert catagorical variables into dummy values.
X=pd.get_dummies(X)
X
```

Out[37]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Loan_ID_LP001002	Loan_ID_LP001003	Loan_ID_LP00
0	5849	0.0	35.0	360.0	1.0	1	0	
1	4583	1508.0	128.0	360.0	1.0	0	1	
2	3000	0.0	66.0	360.0	1.0	0	0	
3	2583	2358.0	120.0	360.0	1.0	0	0	
4	6000	0.0	141.0	360.0	1.0	0	0	
...	...	...	...	...	...	...	...	
609	2900	0.0	71.0	360.0	1.0	0	0	
610	4106	0.0	40.0	180.0	1.0	0	0	
611	8072	240.0	253.0	360.0	1.0	0	0	
612	7583	0.0	187.0	360.0	1.0	0	0	
613	4583	0.0	133.0	360.0	0.0	0	0	

614 rows × 640 columns

In [38]:
```
#Training and Testing of data:-It is used To predict the Outcome of our model to get accurate results.
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=42)
```

In [39]:
```
X_train.head(5)            #prints the top 5 values from the X_train.
```

Out[39]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Loan_ID_LP001002	Loan_ID_LP001003	Loan_ID_LP00
83	6000	2250.0	265.0	360.0	35.0	0	0	
90	2958	2900.0	131.0	360.0	1.0	0	0	
227	6250	1695.0	210.0	360.0	1.0	0	0	
482	2083	3150.0	128.0	360.0	1.0	0	0	
464	4166	0.0	98.0	360.0	0.0	0	0	

5 rows × 640 columns

In [40]:
```
y_train.head(5)                         #prints the top 5 values from the y_train.
```

Out[40]:
```
83     N
90     Y
227    Y
482    Y
464    N
Name: Loan_Status, dtype: object
```

(7)-Build the machine learning model which ever is suitable for the dataset

In [42]:
```
#Importing the RandomForestClassifier Model for classification and regression.
from sklearn.ensemble import RandomForestClassifier

model = RandomForestClassifier()
```

(8)-Fitting the model on the training dataset

In [43]:
```
model.fit(X_train, y_train)          #Fitting the model using "fit" Command.
```

Out[43]:
```
RandomForestClassifier()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

(9)-Testing the model and finding the accuracy of the model on the test and the training datasets

In [44]:
```
#To find the Training accuracy and Testing accuracy.
from sklearn.metrics import accuracy_score

y_pred_train = model.predict(X_train)
y_pred_test = model.predict(X_test)

train_accuracy = accuracy_score(y_train, y_pred_train)
test_accuracy = accuracy_score(y_test, y_pred_test)

print("Training Accuracy:", train_accuracy)
print("Test Accuracy:", test_accuracy)
```

```
Training Accuracy: 1.0
Test Accuracy: 0.7886178861788617
```

In [52]:
```
#Predict:- To Predict the outcome of our model.
ypred=model.predict(X_test)
ypred
```

Out[52]:
```
array(['Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'N', 'N', 'Y', 'Y',
       'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'N', 'Y', 'N', 'Y', 'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'N',
       'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y',
       'N', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
       'Y', 'N', 'Y', 'N', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'Y'], dtype=object)
```

In [46]:
```
model.score(X_test,y_test)      #Testing.
```

Out[46]:
```
0.7886178861788617
```

In [47]:
```
model.score(X_test,ypred)      #Accuracy & Prediction.
```

Out[47]:
```
1.0
```

In [ ]:

(10)-Creating a confusion matrix

```
#Confusion Matrix:it is a table that is often used for describe the performance of your classification model on the
set of data for which the values are known.
```

In [49]:
```
from sklearn.metrics import confusion_matrix

confusion_matrix_test = confusion_matrix(y_test, y_pred_test)
print("Confusion Matrix (Test Data):\n", confusion_matrix_test)
```

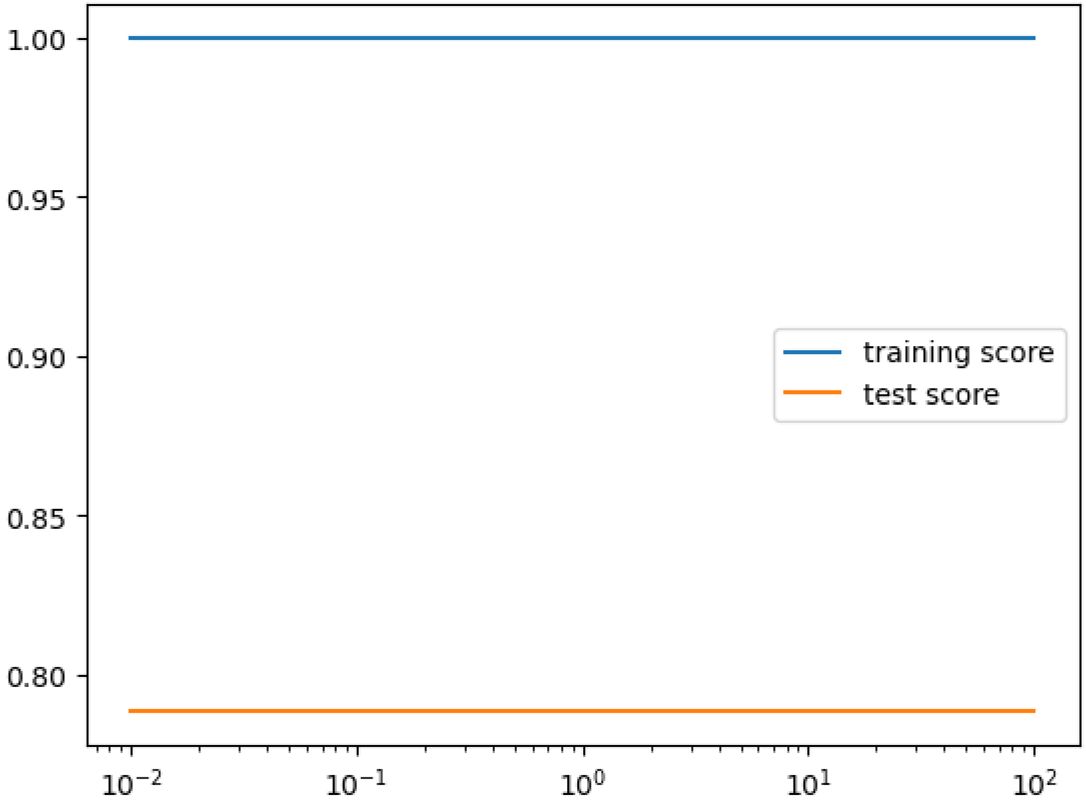
```
Confusion Matrix (Test Data):
 [[18 25]
 [ 1 79]]
```

CONCLUSION:-

In [50]:
```
#To plot the accuracy b/n Training and Testing using MatplotLibrary.
c = [0.01,0.1,1,10,100]
test_score=[]
train_score=[]
for i in c:
    clf =RandomForestClassifier()
    clf.fit(X_train,y_train)
    train_score.append(clf.score(X_train,y_train))
    test_score.append(clf.score(X_test,y_test))

plt.plot(c, train_score, label="training score")
plt.plot(c, test_score, label="test score")
plt.xscale('log')
plt.legend()
```

Out[50]:
```
<matplotlib.legend.Legend at 0x2ce4967a310>
```


1.00
0.95
0.90
0.85
0.80
training score
test score
10^{-2}
10^{-1}
10^{0}
10^{1}
10^{2}

In [ ]: