In [746]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

In [747]:
```
import io
%cd "C:\Users\abhinav\Desktop\Analytics\Loan Predicition"
```

```
C:\Users\abhinav\Desktop\Analytics\Loan Predicition
```

In [748]:
```
traindata=pd.read_csv("train_ctrUa4K.csv")
```

In [749]:
```
testdata=pd.read_csv("test_lAUu6dG.csv")
```

In [750]:
```
traindata.shape
```

Out[750]:
```
(614, 13)
```

In [751]:
```
testdata.shape
```

Out[751]:
```
(367, 12)
```

In [752]:
```
traindata.dtypes #no duplicate columns or variables
```

Out[752]:
```
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
```

In [753]:

```
traindata.describe()
#no zero values or single value column
```

Out[753]:

	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 [754]:

```
testdata.dtypes
```

Out[754]:

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

In [755]:

```
#check for missing values and NA
traindata.isnull().sum().sort_values(ascending=False)
```

Out[755]:

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

In [756]:
```
testdata.isnull().sum().sort_values(ascending=False)
```

Out[756]:
```
Credit_History       29
Self_Employed        23
Gender               11
Dependents           10
Loan_Amount_Term      6
LoanAmount            5
Property_Area         0
CoapplicantIncome     0
ApplicantIncome       0
Education             0
Married               0
Loan_ID               0
dtype: int64
```

In [757]:
```
#clean train data
traindata.Credit_History.value_counts(dropna=False)
```

Out[757]:
```
1.0    475
0.0     89
NaN     50
Name: Credit_History, dtype: int64
```

In [758]:
```
#impute with not available
traindata.Credit_History=traindata.Credit_History.fillna(traindata.Credit_His
```

In [759]:
```
traindata.Self_Employed.value_counts(dropna=False)
```

Out[759]:
```
No     500
Yes     82
NaN     32
Name: Self_Employed, dtype: int64
```

In [760]:
```
traindata.Self_Employed=traindata.Self_Employed.fillna("NotAvailable")
```

In [761]:
```
traindata.LoanAmount.describe()
```

Out[761]:
```
count    592.000000
mean     146.412162
std       85.587325
min        9.000000
25%      100.000000
50%      128.000000
75%      168.000000
max      700.000000
Name: LoanAmount, dtype: float64
```

In [762]:
```
traindata.LoanAmount=traindata.LoanAmount.fillna(traindata.LoanAmount.mean())
```

In [763]:

```
traindata.Dependents.value_counts(dropna=False)
```

Out[763]:

```
0      345
1      102
2      101
3+      51
NaN     15
Name: Dependents, dtype: int64
```

In [764]:

```
traindata.Dependents=traindata.Dependents.fillna(traindata.Dependents.value_c
```

In [765]:

```
traindata.Loan_Amount_Term.value_counts(dropna=False)
```

Out[765]:

```
360.0    512
180.0     44
480.0     15
NaN       14
300.0     13
84.0       4
240.0      4
120.0      3
36.0       2
60.0       2
12.0       1
Name: Loan_Amount_Term, dtype: int64
```

In [766]:

```
traindata.Loan_Amount_Term=traindata.Loan_Amount_Term.fillna(traindata.Loan_A
```

In [767]:

```
traindata.Gender.value_counts(dropna=False)
```

Out[767]:

```
Male      489
Female    112
NaN        13
Name: Gender, dtype: int64
```

In [768]:

```
traindata.Gender=traindata.Gender.fillna("NotAvailabe")
```

In [769]:

```
traindata.Married.value_counts(dropna=False)
```

Out[769]:

```
Yes    398
No     213
NaN      3
Name: Married, dtype: int64
```

In [770]:

```
traindata.Married=traindata.Married.fillna("NotAvailable") #All train data is
```

In [771]:

```
traindata.shape
```

Out[771]:

```
(614, 13)
```

In [772]:
```
testdata.isnull().sum().sort_values(ascending=False)
```

Out[772]:
```
Credit_History       29
Self_Employed        23
Gender               11
Dependents           10
Loan_Amount_Term      6
LoanAmount            5
Property_Area         0
CoapplicantIncome     0
ApplicantIncome       0
Education             0
Married               0
Loan_ID               0
dtype: int64
```

In [773]:
```
testdata.Gender.value_counts(dropna=False)
```

Out[773]:
```
Male      286
Female     70
NaN        11
Name: Gender, dtype: int64
```

In [774]:
```
testdata.Gender=testdata.Gender.fillna((testdata.Gender.value_counts().idxmax
```

In [775]:
```
testdata.Gender.value_counts()
```

Out[775]:
```
Male      297
Female     70
Name: Gender, dtype: int64
```

In [776]:
```
testdata.Credit_History.value_counts(dropna=False)
```

Out[776]:
```
1.0    279
0.0     59
NaN     29
Name: Credit_History, dtype: int64
```

In [777]:
```
testdata.Credit_History=testdata.Credit_History.fillna((testdata.Credit_Histo
```

In [778]:
```
testdata.Credit_History.value_counts()
```

Out[778]:
```
1.0    308
0.0     59
Name: Credit_History, dtype: int64
```

In [779]:

```
testdata.Self_Employed.value_counts(dropna=False)
```

Out[779]:

```
No     307
Yes     37
NaN     23
Name: Self_Employed, dtype: int64
```

In [780]:

```
testdata.Self_Employed=testdata.Self_Employed.fillna((testdata.Self_Employed.
```

In [781]:

```
testdata.Self_Employed.value_counts()
```

Out[781]:

```
No     330
Yes     37
Name: Self_Employed, dtype: int64
```

In [782]:

```
testdata.Dependents.value_counts(dropna=False)
```

Out[782]:

```
0      200
2       59
1       58
3+      40
NaN     10
Name: Dependents, dtype: int64
```

In [783]:

```
testdata.Dependents=testdata.Dependents.fillna(testdata.Dependents.value_coun
```

In [784]:

```
testdata.Dependents.value_counts()
```

Out[784]:

```
0     210
2      59
1      58
3+     40
Name: Dependents, dtype: int64
```

In [785]:

```
testdata.LoanAmount.value_counts(dropna=False)
```

Out[785]:

```
150.0    12
125.0    11
110.0    10
100.0     9
90.0      9
         ..
186.0     1
163.0     1
360.0     1
412.0     1
297.0     1
Name: LoanAmount, Length: 145, dtype: int64
```

In [786]:

```
testdata.LoanAmount=testdata.LoanAmount.fillna(testdata.LoanAmount.mean())
```

```
In [787]: testdata.LoanAmount.value_counts()

Out[787]: 150.0    12
          125.0    11
          110.0    10
          100.0     9
          90.0      9
                   ..
          225.0     1
          103.0     1
          153.0     1
          199.0     1
          71.0      1
          Name: LoanAmount, Length: 145, dtype: int64
```

```
In [788]: testdata.Loan_Amount_Term.value_counts(dropna=False)

Out[788]: 360.0    311
          180.0     22
          480.0      8
          300.0      7
          NaN        6
          240.0      4
          84.0       3
          6.0        1
          120.0      1
          36.0       1
          350.0      1
          12.0       1
          60.0       1
          Name: Loan_Amount_Term, dtype: int64
```

```
In [789]: testdata.Loan_Amount_Term=testdata.Loan_Amount_Term.fillna(testdata.Loan_Amou
```

```
In [790]: traindata.shape

Out[790]: (614, 13)
```

```
In [791]: traindata.columns

Out[791]: Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
                 'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmoun
          t',
                 'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Statu
          s'],
                dtype='object')
```

In [792]:
```
traindata.dtypes
```

Out[792]:
```
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
```

In [793]:
```
objectcolumns=traindata[['Loan_ID','Gender','Married','Education','Self_Emplo
```

In [794]:
```
numericcolumns=traindata[["ApplicantIncome","CoapplicantIncome","LoanAmount",
```

In [795]:
```
#dumy variable of objectcolumns
from sklearn.preprocessing import LabelEncoder
```

In [796]:
```
le=LabelEncoder()
```

In [797]:
```
objectcolumns=objectcolumns.apply(le.fit_transform)
```

In [798]:
```
objectcolumns.head()
```

Out[798]:

	Loan_ID	Gender	Married	Education	Self_Employed	Property_Area	Loan_Status	Dependents
0	0	1	0	0	0	2	1	0
1	1	1	2	0	0	0	0	1
2	2	1	2	0	2	2	1	0
3	3	1	2	1	0	2	1	0
4	4	1	0	0	0	2	1	0

In [799]:

```
numericcolumns.head()
```

Out[799]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term
0	5849	0.0	146.412162	360.0
1	4583	1508.0	128.000000	360.0
2	3000	0.0	66.000000	360.0
3	2583	2358.0	120.000000	360.0
4	6000	0.0	141.000000	360.0

In [831]:

```
combinedtraindata=pd.concat([numericcolumns,objectcolumns],axis=1)
```

In [832]:

```
testdata.columns
```

Out[832]:

```
Index(['Loan_ID', 'Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmoun
t',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area'],
      dtype='object')
```

In [840]:

```
objectdummy=testdata[['Loan_ID','Gender','Married','Education','Self_Employed
```

In [841]:

```
numericdummy=testdata[["ApplicantIncome","CoapplicantIncome","LoanAmount","Lo
```

In [842]:

```
objectdummy=objectdummy.apply(le.fit_transform)
```

In [843]:

```
objectdummy.head()
```

Out[843]:

	Loan_ID	Gender	Married	Education	Self_Employed	Property_Area	Dependents
0	0	1	1	0	0	2	0
1	1	1	1	0	0	2	1
2	2	1	1	0	0	2	2
3	3	1	1	0	0	2	2
4	4	1	0	1	0	2	0

In [844]:

```
combinedtestdata=pd.concat([numericdummy,objectdummy],axis=1)
```

In [845]:

```
print(combinedtraindata.shape)
print(combinedtestdata.shape)
```

```
(614, 13)
(367, 11)
```

In [847]:

```
Y=combinedtraindata.Loan_Status
X=combinedtraindata.drop(["Loan_Status","Loan_ID"],axis=1)  #split data into
```

In [848]:

```
from sklearn.linear_model import LogisticRegression
```

In [849]:

```
reg=LogisticRegression(max_iter=5000)
```

In [850]:

```
regmodel=reg.fit(X,Y)
```

In [851]:

```
regmodel.score(X,Y)
```

Out[851]:

```
0.8110749185667753
```

In [852]:

```
regpredict=regmodel.predict(X)
```

In [853]:
```
regpredict
```

Out[853]:
```
array([1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1,
       0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1,
       0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1,
       1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1,
       1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 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, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1,
       1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 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, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1,
       1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 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, 0,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1,
       0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1,
       1, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0])
```

In [854]:
```
regtestpredict=regmodel.predict(combinedtestdata)
```

In [855]:
```
regtestpredict
```

Out[855]:
```
array([0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0])
```

In [856]:

```
pd.DataFrame(regtestpredict).to_csv('reg.csv')
```

In [ ]: