In [30]:

```
import pandas as pd
import numpy as np
```

In [57]:

```
#Import the Dataset
df = pd.read_csv('train_ctrUa4K.csv')
```

In [4]:

```
#Checking the Shape of the DataFrame
df.shape
```

Out[4]:

```
(614, 13)
```

In [5]:

```
#Glance at the DataFrame
df.head(5)
```

Out[5]:

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

In [9]:

```
#Checking the Balance of Target Variable
df.Loan_Status.value_counts()
```

Out[9]:

```
Y    422
N    192
Name: Loan_Status, dtype: int64
```

In [62]:

```
#Checking the Missing Fields
df[df.isnull().any(axis=1)].count()
```

Out[62]:

```
Loan_ID               134
Gender                121
Married               131
Dependents            119
Education             134
Self_Employed         102
ApplicantIncome       134
CoapplicantIncome     134
LoanAmount            112
Loan_Amount_Term      120
Credit_History         84
Property_Area         134
Loan_Status           134
dtype: int64
```

In [12]:

```
#Replacing Missing Values with Mode rather than dropping the rows
df['Gender'].fillna(df['Gender'].mode()[0], inplace=True)
df['Married'].fillna(df['Married'].mode()[0], inplace=True)
df['Dependents'].fillna(df['Dependents'].mode()[0], inplace=True)
df['Self_Employed'].fillna(df['Self_Employed'].mode()[0], inplace=True)
df['Credit_History'].fillna(df['Credit_History'].mode()[0], inplace=True)
```

In [13]:

```
df['Loan_Amount_Term'].value_counts()
```

Out[13]:

```
360.0    512
180.0     44
480.0     15
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 [14]:

```
df['Loan_Amount_Term'].fillna(df['Loan_Amount_Term'].mode()[0], inplace=True)
```

In [24]:

```
df['LoanAmount'].fillna(df['LoanAmount'].median(), inplace=True)
```

In [25]:

```
df.Loan_Amount_Term.value_counts()
```

Out[25]:

```
360.0    526
180.0     44
480.0     15
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 [ ]:

```
df['Married'].fillna(df['Married'].mode()[0], inplace=True)
df['Dependents'].fillna(df['Dependents'].mode()[0], inplace=True)
df['Self_Employed'].fillna(df['Self_Employed'].mode()[0], inplace=True)
df['Credit_History'].fillna(df['Credit_History'].mode()[0], inplace=True)
```

In [27]:

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

Out[27]:

```
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
```

In [31]:

```
df['LoanAmount_log'] = np.log(df['LoanAmount'])
```

Out[31]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x117305e48>
```

In [33]:

```
df=df.drop('Loan_ID',axis=1)
```

In [34]:

```
y = df.Loan_Status
X = df.drop('Loan_Status',1)
```

In [45]:

```
#Replacing Categorial Variables with Dummy Variables
X=pd.get_dummies(X)
df=pd.get_dummies(df)
```

In [46]:

```
from sklearn.model_selection import train_test_split
x_train, x_cv, y_train, y_cv = train_test_split(X,y, test_size =0.3)
```

In [47]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
model = LogisticRegression()
model.fit(x_train, y_train)
```

```
/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic
.py:433: FutureWarning: Default solver will be changed to 'lbfgs' in
0.22. Specify a solver to silence this warning.
  FutureWarning)
```

Out[47]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_interce
pt=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=None, solver='warn
',
          tol=0.0001, verbose=0, warm_start=False)
```

In [48]:

```
pred_cv = model.predict(x_cv)
```

In [61]:

```
print('Accuracy Score ', accuracy_score(y_cv,pred_cv))
print('Accuracy of logistic regression classifier on test set: {:.2f}'.format(mod
```

```
Accuracy Score  0.8108108108108109
Accuracy of logistic regression classifier on test set: 0.81
```

In [56]:

```
df.head(5)
```

Out[56]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Loa
0	5849	0.0	128.0	360.0	1.0	
1	4583	1508.0	128.0	360.0	1.0	
2	3000	0.0	66.0	360.0	1.0	
3	2583	2358.0	120.0	360.0	1.0	
4	6000	0.0	141.0	360.0	1.0	

5 rows × 23 columns

In [59]:

```
from sklearn.metrics import classification_report
print(classification_report(y_cv, pred_cv))
```

```
              precision    recall  f1-score   support

           N       0.96      0.44      0.61        61
           Y       0.78      0.99      0.88       124

   micro avg       0.81      0.81      0.81       185
   macro avg       0.87      0.72      0.74       185
weighted avg       0.84      0.81      0.79       185
```

In [60]:

```
from sklearn.metrics import confusion_matrix
confusion_matrix = confusion_matrix(y_cv, pred_cv)
print(confusion_matrix)
```

```
[[ 27  34]
 [  1 123]]
```

In [ ]:

In [30]:

In [57]:

In [4]:

Out[4]:

```
(614, 13)
```

In [5]:

Out[5]:

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

In [9]:

Out[9]:

```
Y    422
N    192
Name: Loan_Status, dtype: int64
```

In [62]:

Out[62]:

```
Loan_ID              134
Gender               121
Married              131
Dependents           119
Education            134
Self_Employed        102
ApplicantIncome      134
CoapplicantIncome    134
LoanAmount           112
Loan_Amount_Term     120
Credit_History        84
Property_Area        134
Loan_Status          134
dtype: int64
```

In [12]:

In [13]:

Out[13]:

```
360.0    512
180.0     44
480.0     15
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 [14]:

In [24]:

In [25]:

Out[25]:

```
360.0    526
180.0     44
480.0     15
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 [ ]:

In [27]:

Out[27]:

```
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
```

In [31]:

Out[31]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x117305e48>
```

In [33]:

In [34]:

In [45]:

In [46]:

In [47]:

```
/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/logistic
.py:433: FutureWarning: Default solver will be changed to 'lbfgs' in
0.22. Specify a solver to silence this warning.
  FutureWarning)
```

Out[47]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_interce
pt=True,
          intercept_scaling=1, max_iter=100, multi_class='warn',
          n_jobs=None, penalty='l2', random_state=None, solver='warn
',
          tol=0.0001, verbose=0, warm_start=False)
```

In [48]:

In [61]:

```
Accuracy Score  0.8108108108108109
Accuracy of logistic regression classifier on test set: 0.81
```

In [56]:

Out[56]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History	Loa
0	5849	0.0	128.0	360.0	1.0	
1	4583	1508.0	128.0	360.0	1.0	
2	3000	0.0	66.0	360.0	1.0	
3	2583	2358.0	120.0	360.0	1.0	
4	6000	0.0	141.0	360.0	1.0	

5 rows × 23 columns

In [59]:

```
              precision    recall  f1-score   support

           N       0.96      0.44      0.61        61
           Y       0.78      0.99      0.88       124

   micro avg       0.81      0.81      0.81       185
   macro avg       0.87      0.72      0.74       185
weighted avg       0.84      0.81      0.79       185
```

In [60]:

```
[[ 27  34]
 [  1 123]]
```

In [ ]: