Loan Prediction

In this model we will predict the eligibilty of a person for loan.

Importing required libraries

In [156]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Importing our training data

In [157]:
```python
train=pd.read_csv('train.csv')
```

In [158]:
```python
train.head()
```

Out[158]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loa
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	

Checking for null values

In [159]:
```python
train.isnull().sum()
```

Out[159]:
```
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
```

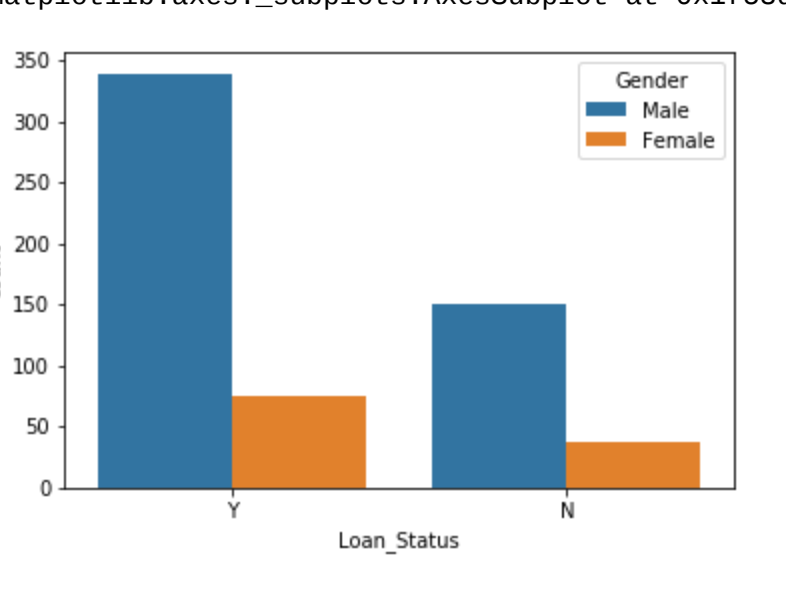
We can see that we have quite a lot null values.Therefore, first we will determine the required features and then deal with the null values.

Finding relation between different features and our target variable

Relation between gender and loan status.

In [160]:
```python
sns.countplot(x='Loan_Status',hue='Gender',data=train)
```

Out[160]: <matplotlib.axes._subplots.AxesSubplot at 0x1f53d072e88>

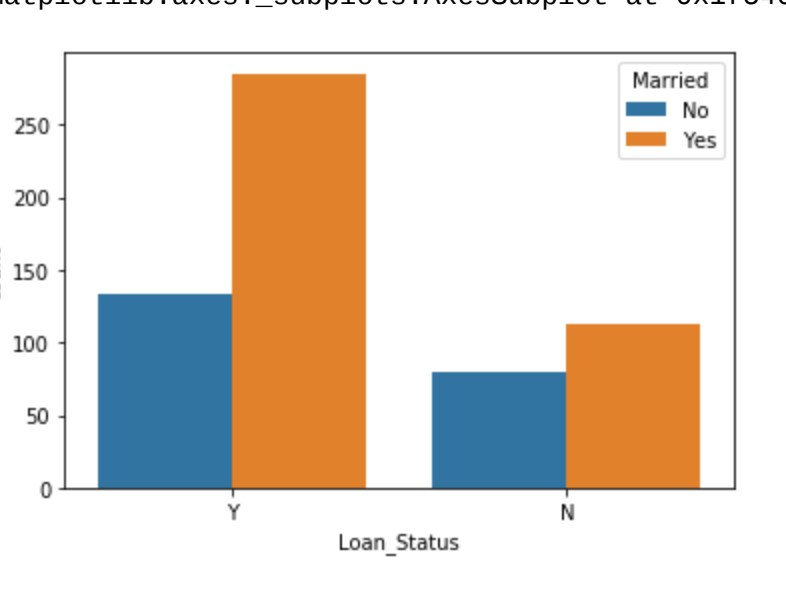


We see that the gender of the applicant does not play a significant role in the process.

Relation between relationship status and loan status.

In [161]:
```python
sns.countplot(x='Loan_Status',hue='Married',data=train)
```

Out[161]: <matplotlib.axes._subplots.AxesSubplot at 0x1f540ceb148>

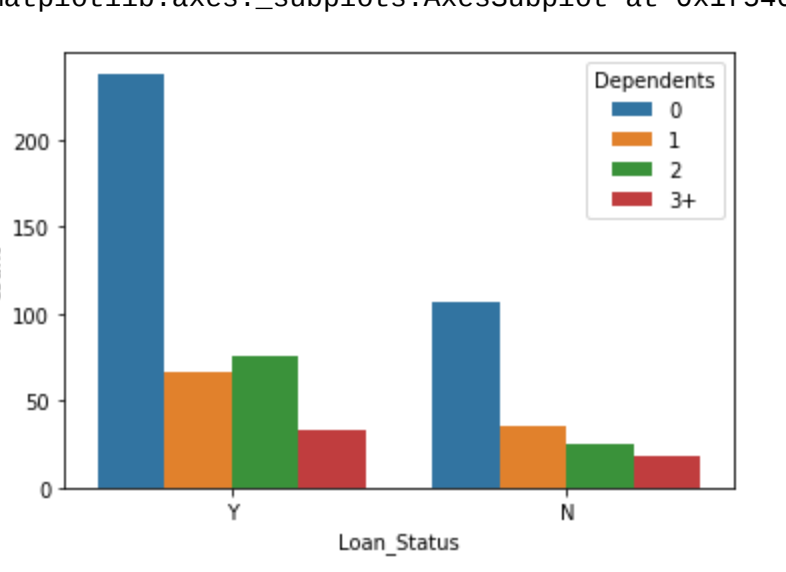


We see that married people have a greater chance of getting the loan.

Relation between dependents and loan status.

In [162]:
```python
sns.countplot(x='Loan_Status',hue='Dependents',data=train)
```

Out[162]: <matplotlib.axes._subplots.AxesSubplot at 0x1f540cf5a88>

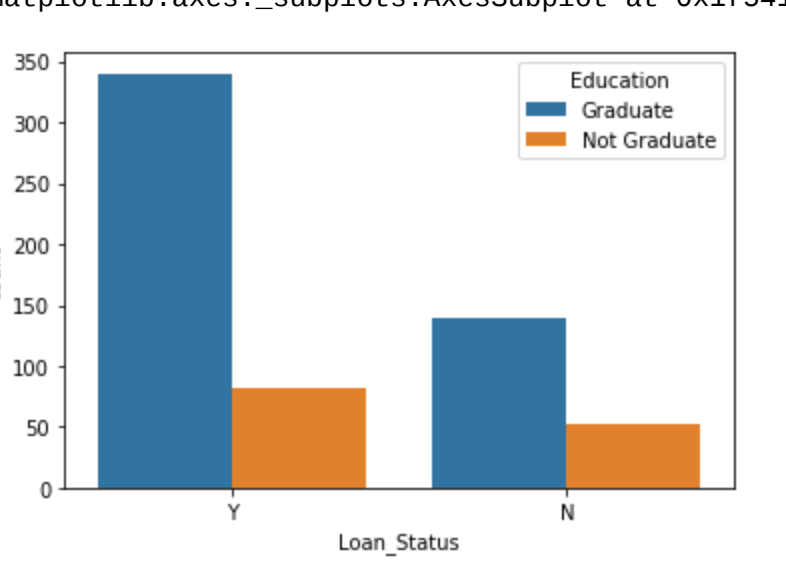


People with 0 dependents have more chances of getting the loan.

Relation between education level of the applicant and loan status.

In [163]:
```python
sns.countplot(x='Loan_Status',hue='Education',data=train)
```

Out[163]: <matplotlib.axes._subplots.AxesSubplot at 0x1f5414aa948>

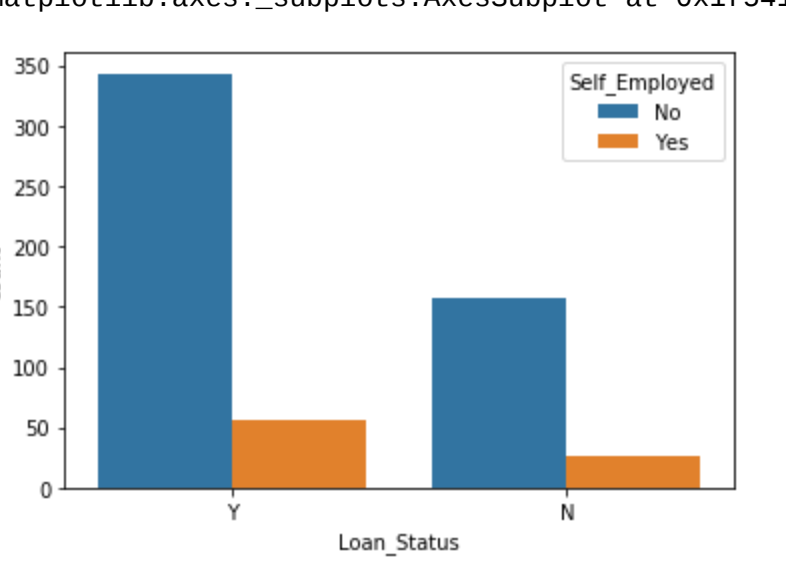


Graduates have more chances of getting the loan.

Relation between the applicant's employment type and loan status.

In [164]:
```python
sns.countplot(x='Loan_Status',hue='Self_Employed',data=train)
```

Out[164]: <matplotlib.axes._subplots.AxesSubplot at 0x1f54150ee48>

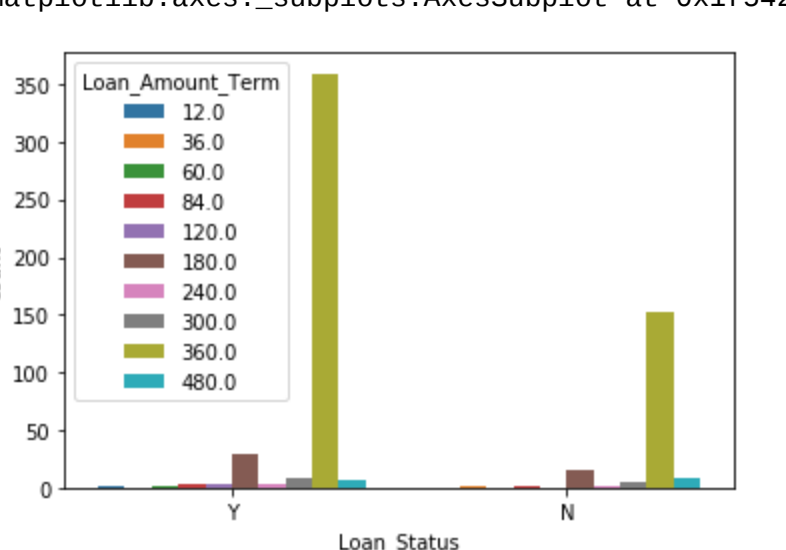


People who are not self employed have higher chances of getting the loan.

Relation between loan amount term and loan status.

In [168]:
```python
sns.countplot(x='Loan_Status',hue='Loan_Amount_Term',data=train)
```

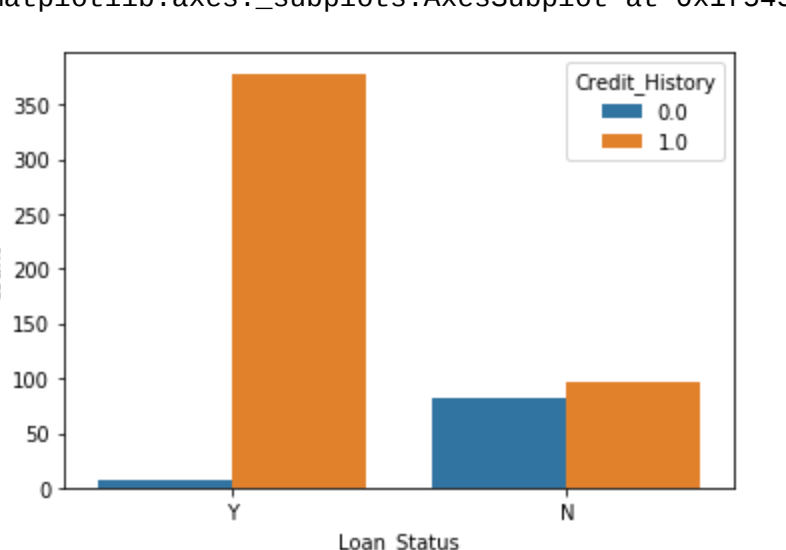
Out[168]: <matplotlib.axes._subplots.AxesSubplot at 0x1f54283b708>



Relation between credit history and loan status.

In [169]:
```python
sns.countplot(x='Loan_Status',hue='Credit_History',data=train)
```

Out[169]: <matplotlib.axes._subplots.AxesSubplot at 0x1f545c47b08>

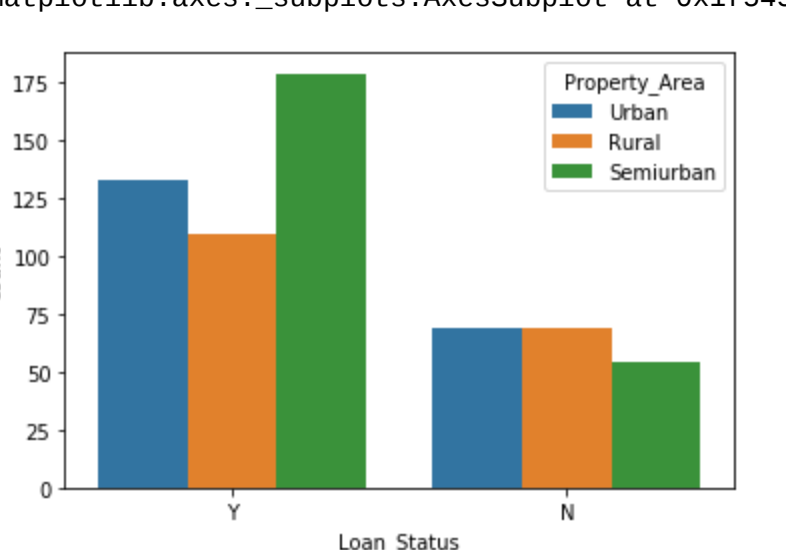


Credit history play a significant role in the process.

Relation between property area and loan status.

In [170]:
```python
sns.countplot(x='Loan_Status',hue='Property_Area',data=train)
```

Out[170]: <matplotlib.axes._subplots.AxesSubplot at 0x1f545c479c8>



Whether the applicant will get the loan or not does not depend on his/her property area.

Filtering out required features

In [171]:
```python
train.drop(['Gender','CoapplicantIncome','Loan_Amount_Term','Property_Area'],axis=1,inplace=True);
```

In [172]:
```python
train.head()
```

Out[172]:

	Loan_ID	Married	Dependents	Education	Self_Employed	ApplicantIncome	LoanAmount	Credit_History	Loan_Status
0	LP001002	No	0	Graduate	No	5849	NaN	1.0	Y
1	LP001003	Yes	1	Graduate	No	4583	128.0	1.0	N
2	LP001005	Yes	0	Graduate	Yes	3000	66.0	1.0	Y
3	LP001006	Yes	0	Not Graduate	No	2583	120.0	1.0	Y
4	LP001008	No	0	Graduate	No	6000	141.0	1.0	Y

Converting categorical data into indicator data

In [173]:
```python
married=pd.get_dummies(train['Married'],drop_first=True)
education=pd.get_dummies(train['Education'],drop_first=True)
self_employed=pd.get_dummies(train['Self_Employed'],drop_first=True)
```

In [174]:
```python
train=pd.concat([train,married,education],axis=1)
train.head()
```

Out[174]:

	Loan_ID	Married	Dependents	Education	Self_Employed	ApplicantIncome	LoanAmount	Credit_History	Loan_Status	Ye
0	LP001002	No	0	Graduate	No	5849	NaN	1.0	Y	
1	LP001003	Yes	1	Graduate	No	4583	128.0	1.0	N	
2	LP001005	Yes	0	Graduate	Yes	3000	66.0	1.0	Y	
3	LP001006	Yes	0	Not Graduate	No	2583	120.0	1.0	Y	
4	LP001008	No	0	Graduate	No	6000	141.0	1.0	Y	

In [175]:
```python
train.rename(columns={'Yes':'married'},inplace=True)
train.head()
```

Out[175]:

	Loan_ID	Married	Dependents	Education	Self_Employed	ApplicantIncome	LoanAmount	Credit_History	Loan_Status	ma
0	LP001002	No	0	Graduate	No	5849	NaN	1.0	Y	
1	LP001003	Yes	1	Graduate	No	4583	128.0	1.0	N	
2	LP001005	Yes	0	Graduate	Yes	3000	66.0	1.0	Y	
3	LP001006	Yes	0	Not Graduate	No	2583	120.0	1.0	Y	
4	LP001008	No	0	Graduate	No	6000	141.0	1.0	Y	

In [176]:
```python
train=pd.concat([train,self_employed],axis=1)
train.head()
```

Out[176]:

	Loan_ID	Married	Dependents	Education	Self_Employed	ApplicantIncome	LoanAmount	Credit_History	Loan_Status	ma
0	LP001002	No	0	Graduate	No	5849	NaN	1.0	Y	
1	LP001003	Yes	1	Graduate	No	4583	128.0	1.0	N	
2	LP001005	Yes	0	Graduate	Yes	3000	66.0	1.0	Y	
3	LP001006	Yes	0	Not Graduate	No	2583	120.0	1.0	Y	
4	LP001008	No	0	Graduate	No	6000	141.0	1.0	Y	

In [177]:
```python
train.rename(columns={'Yes':'self_employed'},inplace=True)
train.head()
```

Out[177]:

	Loan_ID	Married	Dependents	Education	Self_Employed	ApplicantIncome	LoanAmount	Credit_History	Loan_Status	ma
0	LP001002	No	0	Graduate	No	5849	NaN	1.0	Y	
1	LP001003	Yes	1	Graduate	No	4583	128.0	1.0	N	
2	LP001005	Yes	0	Graduate	Yes	3000	66.0	1.0	Y	
3	LP001006	Yes	0	Not Graduate	No	2583	120.0	1.0	Y	
4	LP001008	No	0	Graduate	No	6000	141.0	1.0	Y	

In [178]:
```python
train.drop(['Married','Education','Self_Employed'],axis=1,inplace=True)
```

In [179]:
```python
train.head()
```

Out[179]:

	Loan_ID	Dependents	ApplicantIncome	LoanAmount	Credit_History	Loan_Status	married	Not Graduate	self_employed
0	LP001002	0	5849	NaN	1.0	Y	0	0	0
1	LP001003	1	4583	128.0	1.0	N	1	0	0
2	LP001005	0	3000	66.0	1.0	Y	1	0	1
3	LP001006	0	2583	120.0	1.0	Y	1	1	0
4	LP001008	0	6000	141.0	1.0	Y	0	0	0

In [180]:
```python
train.isnull().sum()
```

Out[180]:
```
Loan_ID             0
Dependents         15
ApplicantIncome     0
LoanAmount         22
Credit_History     50
Loan_Status         0
married             0
Not Graduate        0
self_employed       0
dtype: int64
```

We see that dependents is also categorical data.Therefore,we need to convert it to indicator data.

In [181]:
```python
dependents=pd.get_dummies(train['Dependents'],drop_first=True)
```

In [182]:
```python
train=pd.concat([train,dependents],axis=1)
train.head()
```

Out[182]:

	Loan_ID	Dependents	ApplicantIncome	LoanAmount	Credit_History	Loan_Status	married	Not Graduate	self_employed	1	2
0	LP001002	0	5849	NaN	1.0	Y	0	0	0	0	
1	LP001003	1	4583	128.0	1.0	N	1	0	0	1	
2	LP001005	0	3000	66.0	1.0	Y	1	0	1	0	
3	LP001006	0	2583	120.0	1.0	Y	1	1	0	0	
4	LP001008	0	6000	141.0	1.0	Y	0	0	0	0	

In [183]:
```python
train.drop('Dependents',axis=1,inplace=True)
train.head()
```

Out[183]:

	Loan_ID	ApplicantIncome	LoanAmount	Credit_History	Loan_Status	married	Not Graduate	self_employed	1	2	3+
0	LP001002	5849	NaN	1.0	Y	0	0	0	0	0	0
1	LP001003	4583	128.0	1.0	N	1	0	0	1	0	0
2	LP001005	3000	66.0	1.0	Y	1	0	1	0	0	0
3	LP001006	2583	120.0	1.0	Y	1	1	0	0	0	0
4	LP001008	6000	141.0	1.0	Y	0	0	0	0	0	0

Checking for null values

In [184]:
```python
train.isnull().sum()
```

Out[184]:
```
Loan_ID             0
ApplicantIncome     0
LoanAmount         22
Credit_History     50
Loan_Status         0
married             0
Not Graduate        0
self_employed       0
1                   0
2                   0
3+                  0
dtype: int64
```

Filling up null values

In [185]:
```python
def fill_loan(cols):
    amount=cols[0]

    if pd.isnull(amount):
        return train['LoanAmount'].mean()

    return amount
```

In [186]:
```python
train['LoanAmount']=train[['LoanAmount']].apply(fill_loan,axis=1)
```

In [187]:
```python
train.isnull().sum()
```

Out[187]:
```
Loan_ID             0
ApplicantIncome     0
LoanAmount          0
Credit_History     50
Loan_Status         0
married             0
Not Graduate        0
self_employed       0
1                   0
2                   0
3+                  0
dtype: int64
```

In [188]:
```python
def fill_credithistory(cols):
    credit_history=cols[0]

    if pd.isnull(credit_history):
        return 1

    return credit_history
```

In [189]:
```python
train['Credit_History']=train[['Credit_History']].apply(fill_credithistory,axis=1)
```

In [190]:
```python
train.isnull().sum()
```

Out[190]:
```
Loan_ID            0
ApplicantIncome    0
LoanAmount         0
Credit_History     0
Loan_Status        0
married            0
Not Graduate       0
self_employed      0
1                  0
2                  0
3+                 0
dtype: int64
```

We have successfully removed all null values.

Building model

Importing required libraries.

In [191]:
```python
from sklearn.tree import DecisionTreeClassifier
```

In [192]:
```python
classifier=DecisionTreeClassifier()
```

Splitting our data

Separating our features and target variable.

In [193]:
```python
x=train.drop(['Loan_Status','Loan_ID'],axis=1)
y=train['Loan_Status']
```

Importing required libraries.

In [194]:
```python
from sklearn.model_selection import train_test_split
```

In [195]:
```python
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)
```

Training and testing our model

In [196]:
```python
classifier.fit(x_train,y_train)
```

Out[196]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

In [197]:
```python
y_hat=classifier.predict(x_test)
```

In [198]:
```python
y_hat
```

Out[198]:
```
array(['Y', 'N', 'N', 'Y', 'Y', 'N', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'N', 'Y', 'N', 'Y', 'Y', 'N',
       'N', 'Y', 'Y', 'N', 'Y', 'N', 'Y', 'N', 'Y', 'N', 'Y', 'N', 'Y',
       'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y',
       'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'N', 'N', 'N', 'Y', 'Y',
       'Y', 'N', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N',
       'N', 'Y', 'N', 'Y', 'N', 'N', 'Y', 'N', 'N', 'Y', 'Y', 'N', 'Y',
       'N', 'Y', 'Y', 'Y', 'N', 'N', 'N', 'Y', 'Y', 'N', 'N', 'Y', 'Y',
       'Y', 'Y', 'N', 'N', 'N', 'Y', 'Y', 'N', 'Y', 'Y', 'N', 'Y', 'N',
       'Y', 'Y', 'Y', 'Y', 'Y', 'N'], dtype=object)
```

Finding out accuracy score

In [199]:
```python
from sklearn.metrics import accuracy_score
```

In [200]:
```python
accuracy=accuracy_score(y_test,y_hat)
accuracy
```

Out[200]: 0.7073170731707317

Accuracy is quite high.

Importing testing data

In [201]:
```python
test=pd.read_csv('test.csv')
```

In [202]:
```python
test.head()
```

Out[202]:

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

Modifying testing data

In [203]:
```python
test.drop(['Gender','CoapplicantIncome','Loan_Amount_Term','Property_Area'],axis=1,inplace=True);
```

In [204]:
```python
married=pd.get_dummies(test['Married'],drop_first=True)
education=pd.get_dummies(test['Education'],drop_first=True)
self_employed=pd.get_dummies(test['Self_Employed'],drop_first=True)
```

In [205]:
```python
test=pd.concat([test,married,education],axis=1)
```

In [206]:
```python
test.rename(columns={'Yes':'married'},inplace=True)
test.head()
```

Out[206]:

	Loan_ID	Married	Dependents	Education	Self_Employed	ApplicantIncome	LoanAmount	Credit_History	married	No Gradua
0	LP001015	Yes	0	Graduate	No	5720	110.0	1.0	1	
1	LP001022	Yes	1	Graduate	No	3076	126.0	1.0	1	
2	LP001031	Yes	2	Graduate	No	5000	208.0	1.0	1	
3	LP001035	Yes	2	Graduate	No	2340	100.0	NaN	1	
4	LP001051	No	0	Not Graduate	No	3276	78.0	1.0	0	

In [207]:
```python
test=pd.concat([test,self_employed],axis=1)
```

In [208]:
```python
test.rename(columns={'Yes':'self_employed'},inplace=True)
test.head()
```

Out[208]:

	Loan_ID	Married	Dependents	Education	Self_Employed	ApplicantIncome	LoanAmount	Credit_History	married	No Gradua
0	LP001015	Yes	0	Graduate	No	5720	110.0	1.0	1	
1	LP001022	Yes	1	Graduate	No	3076	126.0	1.0	1	
2	LP001031	Yes	2	Graduate	No	5000	208.0	1.0	1	
3	LP001035	Yes	2	Graduate	No	2340	100.0	NaN	1	
4	LP001051	No	0	Not Graduate	No	3276	78.0	1.0	0	

In [209]:
```python
test.drop(['Married','Education','Self_Employed'],axis=1,inplace=True)
```

In [210]:
```python
dependents=pd.get_dummies(test['Dependents'],drop_first=True)
```

In [211]:
```python
test=pd.concat([test,dependents],axis=1)
```

In [212]:
```python
test.drop('Dependents',axis=1,inplace=True)
test.head()
```

Out[212]:

	Loan_ID	ApplicantIncome	LoanAmount	Credit_History	married	Not Graduate	self_employed	1	2	3+
0	LP001015	5720	110.0	1.0	1	0	0	0	0	0
1	LP001022	3076	126.0	1.0	1	0	0	1	0	0
2	LP001031	5000	208.0	1.0	1	0	0	0	1	0
3	LP001035	2340	100.0	NaN	1	0	0	0	1	0
4	LP001051	3276	78.0	1.0	0	1	0	0	0	0

In [213]:
```python
test.isnull().sum()
```

Out[213]:
```
Loan_ID             0
ApplicantIncome     0
LoanAmount          5
Credit_History     29
married             0
Not Graduate        0
self_employed       0
1                   0
2                   0
3+                  0
dtype: int64
```

In [214]:
```python
def fill_loan_test(cols):
    amount=cols[0]

    if pd.isnull(amount):
        return test['LoanAmount'].mean()

    return amount
```

In [215]:
```python
test['LoanAmount']=test[['LoanAmount']].apply(fill_loan_test,axis=1)
```

In [216]:
```python
test['Credit_History']=test[['Credit_History']].apply(fill_credithistory,axis=1)
```

In [217]:
```python
test.isnull().sum()
```

Out[217]:
```
Loan_ID            0
ApplicantIncome    0
LoanAmount         0
Credit_History     0
married            0
Not Graduate       0
self_employed      0
1                  0
2                  0
3+                 0
dtype: int64
```

Making predictions on testing data

In [218]:
```python
x=test.drop(['Loan_ID'],axis=1)
```

In [219]:
```python
predictions=classifier.predict(x)
```

In [220]:
```python
predictions
```

Out[220]:
```
array(['Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'N', 'Y', 'Y',
       'N', 'Y', 'Y', 'Y', 'Y', 'N', 'N', 'Y', 'N', 'Y', 'Y', 'Y', 'Y',
       'Y', 'N', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'N', 'Y', 'Y', 'N', 'Y',
       'Y', 'N', 'Y', 'Y', 'N', 'Y', 'Y', 'N', 'Y', 'N', 'N', 'Y', 'N',
       'Y', 'Y', 'Y', 'Y', 'N', 'N', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y',
       'Y', 'N', 'Y', 'N', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y',
       'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'N', 'N', 'Y', 'Y', 'N', 'N',
       'N', 'N', 'Y', 'N', 'Y', 'N', 'N', 'N', 'Y', 'N', 'Y', 'N', 'Y',
       'Y', 'N', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'N',
       'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'N',
       'Y', 'Y', 'N', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'N', 'N', 'N',
       'N', 'Y', 'Y', 'Y', 'N', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
       'N', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'N', 'N', 'Y',
       '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',
       'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'N',
       'Y', 'N', 'N', 'Y', 'Y', 'N', 'Y', 'N', 'Y', 'Y', 'Y', 'N', 'Y',
       'Y', 'N', 'Y', 'N', 'N', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y',
       'N', 'Y', 'N', 'Y', 'Y', 'N', 'N', 'Y', 'N', 'N', 'Y', 'Y', 'N',
       'Y', 'N', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'N',
       'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'N', 'Y', 'N', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'N', 'Y',
       'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'N', 'N', 'Y', 'Y',
       'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'N', 'Y', 'Y',
       'Y', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'Y', 'N', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y', 'Y',
       'Y', 'Y', 'N'], dtype=object)
```

In [221]:
```python
predictions.size
```

Out[221]: 367

In [222]:
```python
submission=pd.DataFrame({
    'Loan_ID':test['Loan_ID'],
    'Loan_Status':predictions
})
submission.to_csv('submission.csv',index=False)
```