In [2]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:

```
df=pd.read_csv(r"C:\Users\dtdee\OneDrive\Desktop\Letsupgrade_Python\Data_Analysis_Visualisation\loan_data_se
```

In [4]:

```
df.head()
```

Out[4]:

rried	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term
No	0	Graduate	No	5849	0.0	NaN	360.0
Yes	1	Graduate	No	4583	1508.0	128.0	360.0
Yes	0	Graduate	Yes	3000	0.0	66.0	360.0
Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0
No	0	Graduate	No	6000	0.0	141.0	360.0

In [5]:

```
# df.drop('RowNumber',axis=1,inplace=True)
```

In [6]:

```
df.shape
```

Out[6]:

```
(614, 13)
```

In [7]:

```
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             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 [8]:

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

Out[8]:

```
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 [9]:

```
df.head()
```

Out[9]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAn
0	LP001002	Male	No	0	Graduate	No	5849	0.0	
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	
4	LP001008	Male	No	0	Graduate	No	6000	0.0	

In [10]:

```
df.fillna({'Gender':'Female','Married':'No','Self_Employed':'Yes','LoanAmount':df.LoanAmount.mean(),'Loan_A
```

In [11]:

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

Out[11]:

```
Loan_ID               0
Gender                0
Married               0
Dependents           15
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 [12]:

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

Out[12]:

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

In [13]:

```
df[df['Dependents']=='3+'].replace(3,inplace=True)
```

Out[13]:

```
Loan_ID              None
Gender               None
Married              None
Dependents           None
Education            None
Self_Employed        None
ApplicantIncome      None
CoapplicantIncome    None
LoanAmount           None
Loan_Amount_Term     None
Credit_History       None
Property_Area        None
Loan_Status          None
dtype: object
```

In [14]:

```
df['Dependents'].replace('3+',3,inplace=True)
```

In [15]:

```
df.Dependents.value_counts()
```

Out[15]:

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

In [16]:

```
df.head()
```

Out[16]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAn
0	LP001002	Male	No	0	Graduate	No	5849	0.0	146.4
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

In [45]:

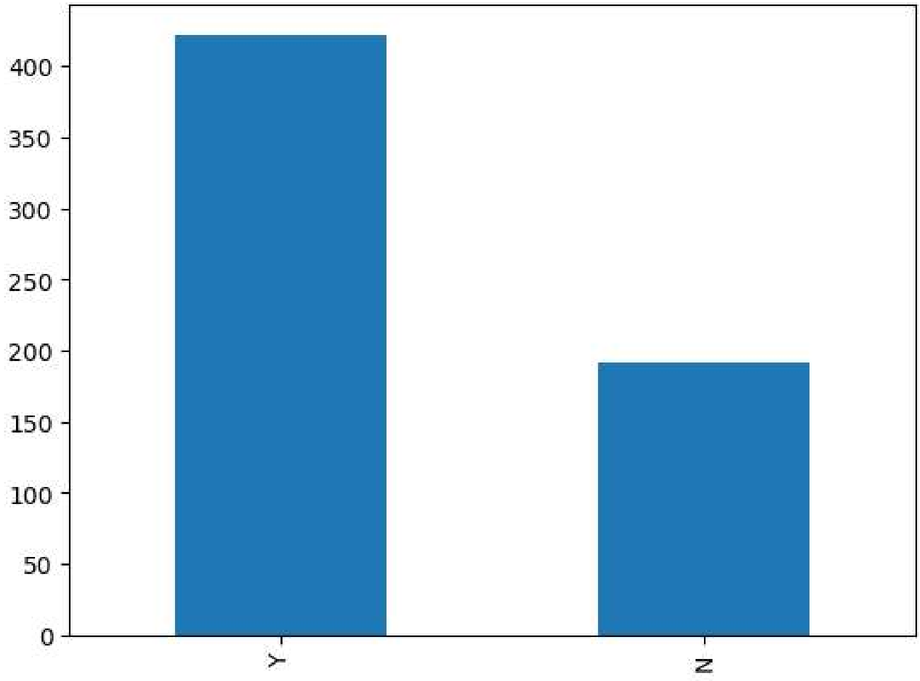
```
#Checking if the dataset is balanced or not

x=df['Loan_Status'].value_counts()
```

In [46]:

```
x.plot(kind='bar')
```

Out[46]:

```
<AxesSubplot:>
```



In [44]:

```
x=df[(df['ApplicantIncome']>20000) & (df['Loan_Status']=='N')]
```

In [19]:

```
p=df.groupby('Dependents')['ApplicantIncome'].mean()
p
```

Out[19]:

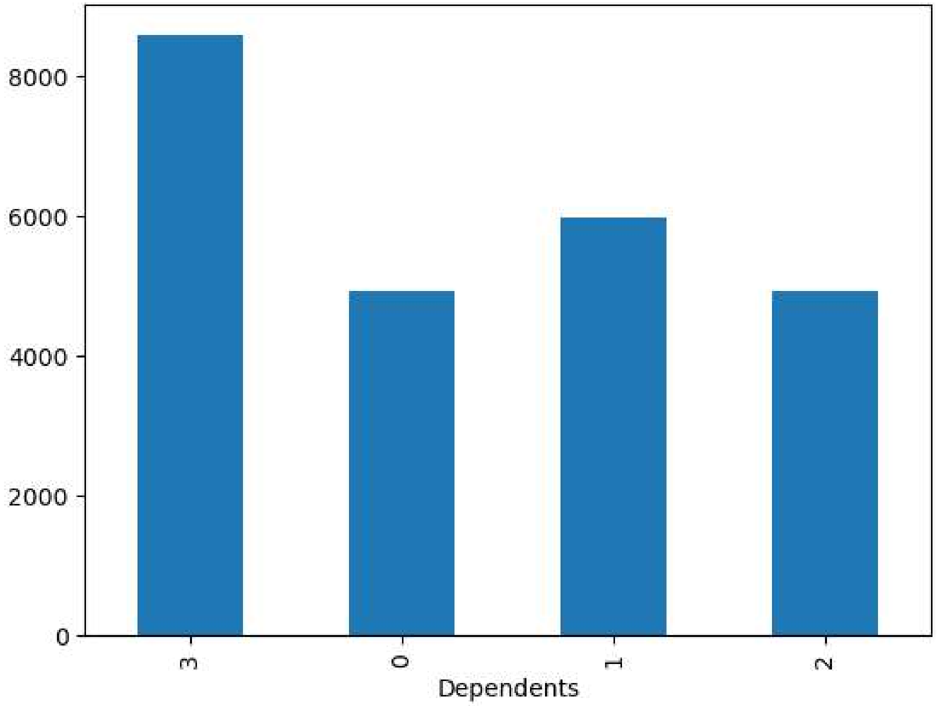
```
Dependents
3    8581.215686
0    4917.423188
1    5962.274510
2    4926.782178
Name: ApplicantIncome, dtype: float64
```

In [20]:

```
p.plot(kind='bar')
```

Out[20]:

```
<AxesSubplot:xlabel='Dependents'>
```

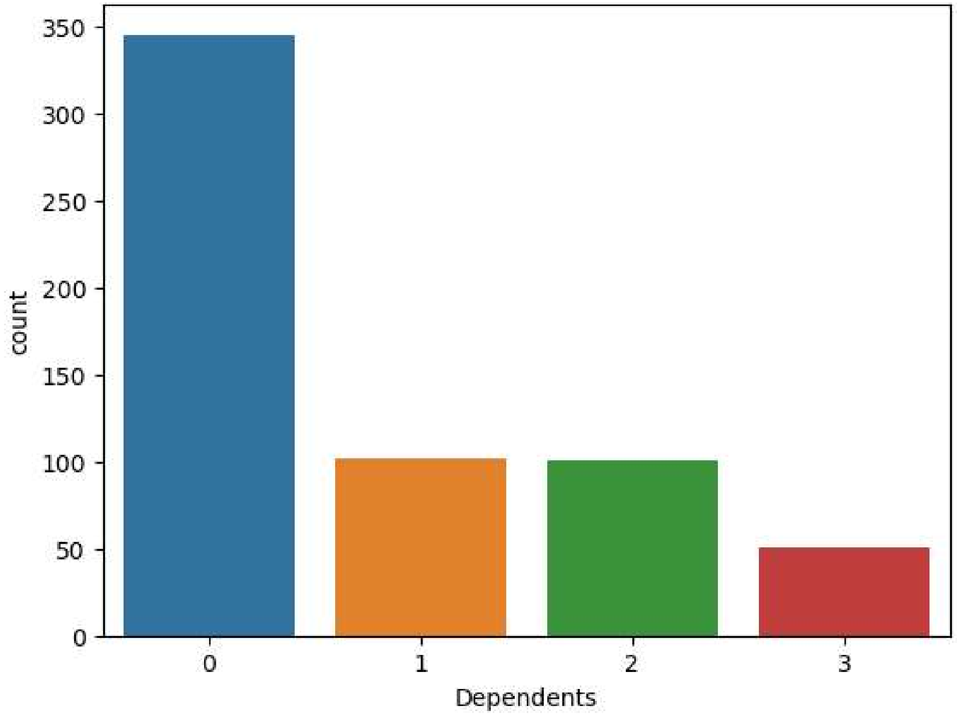


In [21]:

```
sns.countplot(x='Dependents',data=df)
```

Out[21]:

```
<AxesSubplot:xlabel='Dependents', ylabel='count'>
```



In [22]:

```
df.Dependents.value_counts()
```

Out[22]:

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

In [23]:

```
df.head()
```

Out[23]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAn
0	LP001002	Male	No	0	Graduate	No	5849	0.0	146.4
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

In [24]:

```
for i in df.columns:
    if type(df[i][1]) != 'int64':
        if i not in ['Loan_ID','ApplicantIncome','CoapplicantIncome','LoanAmount']:
            print(df[i].unique())
```

```
['Male' 'Female']
['No' 'Yes']
['0' '1' '2' 3 nan]
['Graduate' 'Not Graduate']
['No' 'Yes']
[360.0 120.0 240.0 '360.0' 180.0 60.0 300.0 480.0 36.0 84.0 12.0]
[1.0 0.0 '0.0']
['Urban' 'Rural' 'Semiurban']
['Y' 'N']
```

In [25]:

```
df['Dependents'].fillna(3,inplace=True)
```

In [26]:

```
df['Dependents'].isnull().sum()
```

Out[26]:

```
0
```

In [27]:

```
df.head()
```

Out[27]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAn
0	LP001002	Male	No	0	Graduate	No	5849	0.0	146.4
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

In [28]:

```
df.columns
```

Out[28]:

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

In [29]:

```
df=df[['Gender', 'Married', 'Dependents', 'Education',
       'Self_Employed', 'ApplicantIncome', 'CoapplicantIncome', 'LoanAmount',
       'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'Loan_Status']]
```

In [30]:

```
df.head(2)
```

Out[30]:

ried	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term
No	0	Graduate	No	5849	0.0	146.412162	360.0
Yes	1	Graduate	No	4583	1508.0	128.000000	360.0

In [31]:

```
df['Dependents'].unique()
```

Out[31]:

```
array(['0', '1', '2', 3], dtype=object)
```

In [32]:

```
df['Dependents']=df['Dependents'].replace({'0':0,'1':1,'2':2})
```

In [33]:

```
for i in df.columns:
    if i in ['Gender', 'Married', 'Dependents', 'Education','Self_Employed','Loan_Amount_Term', 'Credit_His
        print(f'Unique elements of column {i} = ',df[i].unique())
```

```
Unique elements of column Gender =  ['Male' 'Female']
Unique elements of column Married =  ['No' 'Yes']
Unique elements of column Dependents =  [0 1 2 3]
Unique elements of column Education =  ['Graduate' 'Not Graduate']
Unique elements of column Self_Employed =  ['No' 'Yes']
Unique elements of column Loan_Amount_Term =  [360.0 120.0 240.0 '360.0' 180.0 60.0 300.0 48
0.0 36.0 84.0 12.0]
Unique elements of column Credit_History =  [1.0 0.0 '0.0']
Unique elements of column Property_Area =  ['Urban' 'Rural' 'Semiurban']
Unique elements of column Loan_Status =  ['Y' 'N']
```

In [34]:

```
df['Loan_Amount_Term']=df['Loan_Amount_Term'].replace({'360.0':360})
df['Credit_History']=df['Credit_History'].replace({'0.0':0})
```

In [35]:

```
for i in df.columns:
    if i in ['Gender', 'Married', 'Dependents', 'Education','Self_Employed','Loan_Amount_Term', 'Credit_His
        print(f'Unique elements of column {i} = ',df[i].unique())
```

```
Unique elements of column Gender =  ['Male' 'Female']
Unique elements of column Married =  ['No' 'Yes']
Unique elements of column Dependents =  [0 1 2 3]
Unique elements of column Education =  ['Graduate' 'Not Graduate']
Unique elements of column Self_Employed =  ['No' 'Yes']
Unique elements of column Loan_Amount_Term =  [360. 120. 240. 180.  60. 300. 480.  36.  84.
12.]
Unique elements of column Credit_History =  [1. 0.]
Unique elements of column Property_Area =  ['Urban' 'Rural' 'Semiurban']
Unique elements of column Loan_Status =  ['Y' 'N']
```

In [54]:

```
df.head(2)
```

Out[54]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loai
0	Male	No	0	Graduate	No	5849	0.0	146.412162	
1	Male	Yes	1	Graduate	No	4583	1508.0	128.000000	

In [68]:

```

```

In [182]:

```
from sklearn.preprocessing import StandardScaler,MinMaxScaler,LabelEncoder
from sklearn.model_selection import train_test_split,cross_val_score
from sklearn.metrics import accuracy_score,classification_report,confusion_matrix,mean_squared_error
from math import sqrt
```

In [71]:

```
df
```

Out[71]:

ried	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term
No	0	Graduate	No	5849	0.0	146.412162	360.0
Yes	1	Graduate	No	4583	1508.0	128.000000	360.0
Yes	0	Graduate	Yes	3000	0.0	66.000000	360.0
Yes	0	Not Graduate	No	2583	2358.0	120.000000	360.0
No	0	Graduate	No	6000	0.0	141.000000	360.0
...	...	...	...	...	...	...	...
No	0	Graduate	No	2900	0.0	71.000000	360.0
Yes	3	Graduate	No	4106	0.0	40.000000	180.0
Yes	1	Graduate	No	8072	240.0	253.000000	360.0
Yes	2	Graduate	No	7583	0.0	187.000000	360.0
No	0	Graduate	Yes	4583	0.0	133.000000	360.0

umns

In [72]:

```
encoder= LabelEncoder()
```

In [73]:

```
for i in df.columns:
    if i in ['Gender','Married','Education','Self_Employed','Property_Area','Loan_Status']:
        df[i]=encoder.fit_transform(df[i])
```

In [75]:

```
df.head(3)
```

Out[75]:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loa
0	1	0	0	0	0	5849	0.0	146.412162	
1	1	1	1	0	0	4583	1508.0	128.000000	
2	1	1	0	0	1	3000	0.0	66.000000	

In [99]:

```
scaler= MinMaxScaler()
```

In [100]:

```
X=df.iloc[:,:-1].values
y=df.iloc[:,11:].values
```

In [101]:

```
X_scaled=scaler.fit_transform(X)
y_scaled=scaler.fit_transform(y)
```

In [267]:

```
y_scaled[0:5]
```

Out[267]:

```
array([[1.],
       [0.],
       [1.],
       [1.],
       [1.]])
```

In [103]:

```
y_scaled.shape
```

Out[103]:

```
(614, 1)
```

In [104]:

```
X_scaled.shape
```

Out[104]:

```
(614, 11)
```

In [105]:

```
X_train,X_test,y_train,y_test= train_test_split(X_scaled,y_scaled,test_size=.2011)
```

In [106]:

```
X_train.shape
```

Out[106]:

```
(490, 11)
```

In [107]:

```
y_train.shape
```

Out[107]:

```
(490, 1)
```

Now we will apply all Algorithms one by one for Supervised learning

Using Logistic Regression Model

In [110]:

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

In [161]:

```
reg=LogisticRegression(max_iter=100,random_state=30)
```

In [162]:

```
reg.fit(X_train,y_train)
```

```
C:\Users\dtdee\anaconda3\lib\site-packages\sklearn\utils\validation.py:1143: DataConversionWa
rning: A column-vector y was passed when a 1d array was expected. Please change the shape of
y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[162]:

```
▾        LogisticRegression
LogisticRegression(random_state=30)
```

In [163]:

```
y_pred= reg.predict(X_test)
```

In [164]:

```
y_pred
```

Out[164]:

```
array([1., 1., 0., 0., 0., 1., 1., 1., 1., 0., 1., 1., 1., 1., 0., 1., 0.,
       0., 1., 1., 0., 0., 1., 1., 1., 1., 0., 1., 1., 1., 1., 1., 1., 0.,
       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., 0., 0., 0., 1., 1., 1., 1., 1.,
       1., 1., 1., 1., 1., 0., 0., 1., 0., 0., 1., 1., 1., 1., 1., 0., 0.,
       0., 1., 1., 1., 1., 1., 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., 1.,
       1., 1., 1., 1., 0.])
```

In [165]:

```
# Now Evaluating the model by using different ways of Evaluation

Accuracy_LogReg=accuracy_score(y_pred,y_test)
Accuracy_LogReg
```

Out[165]:

```
0.782258064516129
```

In [166]:

```
print(classification_report(y_pred,y_test))
```

```
              precision    recall  f1-score   support

         0.0       0.52      0.76      0.62        29
         1.0       0.91      0.79      0.85        95

    accuracy                           0.78       124
   macro avg       0.72      0.77      0.73       124
weighted avg       0.82      0.78      0.79       124
```

In [171]:

```
acc=cross_val_score(reg,X_train,y_train,cv=5)
np.mean(acc)
```

```
C:\Users\dtdee\anaconda3\lib\site-packages\sklearn\utils\validation.py:1143: DataConversionWa
rning: A column-vector y was passed when a 1d array was expected. Please change the shape of
y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\dtdee\anaconda3\lib\site-packages\sklearn\utils\validation.py:1143: DataConversionWa
rning: A column-vector y was passed when a 1d array was expected. Please change the shape of
y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\dtdee\anaconda3\lib\site-packages\sklearn\utils\validation.py:1143: DataConversionWa
rning: A column-vector y was passed when a 1d array was expected. Please change the shape of
y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\dtdee\anaconda3\lib\site-packages\sklearn\utils\validation.py:1143: DataConversionWa
rning: A column-vector y was passed when a 1d array was expected. Please change the shape of
y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\dtdee\anaconda3\lib\site-packages\sklearn\utils\validation.py:1143: DataConversionWa
rning: A column-vector y was passed when a 1d array was expected. Please change the shape of
y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[171]:

```
0.7612244897959185
```

In [185]:

```
# Mean Square error is the mean of squares of difference in actual and predicted values

mean_squared_error(y_test,y_pred)
```

Out[185]:

```
0.21774193548387097
```

In [188]:

```
root_mean_squared_error=sqrt(mean_squared_error(y_test,y_pred))
root_mean_squared_error
```

Out[188]:

```
0.4666282626286914
```

Now Using Naive Bayes Method

In [239]:

```
from sklearn.naive_bayes import GaussianNB

model_gb=GaussianNB()
```

In [240]:

```
model_gb.fit(X_train,y_train)
```

```
C:\Users\dtdee\anaconda3\lib\site-packages\sklearn\utils\validation.py:1143: DataConversionWa
rning: A column-vector y was passed when a 1d array was expected. Please change the shape of
y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[240]:

```
▾ GaussianNB
GaussianNB()
```

In [241]:

```
y_pred=model_gb.predict(X_test)
```

In [242]:

```
accuracy_GB = accuracy_score(y_pred,y_test)
accuracy_GB
```

Out[242]:

```
0.7741935483870968
```

In [279]:

```
print(classification_report(y_pred,y_test))
```

```
              precision    recall  f1-score   support

         0.0       0.52      0.73      0.61        30
         1.0       0.90      0.79      0.84        94

    accuracy                           0.77       124
   macro avg       0.71      0.76      0.73       124
weighted avg       0.81      0.77      0.79       124
```

Now Using the Decision Tree Method

In [284]:

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import plot_tree
```

In [282]:

```
dt=DecisionTreeClassifier(criterion='gini',max_depth=3,random_state=10)
```

In [283]:

```
dt.fit(X_train,y_train)
```

Out[283]:

```
▾               DecisionTreeClassifier
DecisionTreeClassifier(max_depth=3, random_state=10)
```

In [292]:

```
y_pred=dt.predict(X_test)
```

In [294]:

```
accuracy_dt=accuracy_score(y_pred,y_test)
accuracy_dt
```

Out[294]:

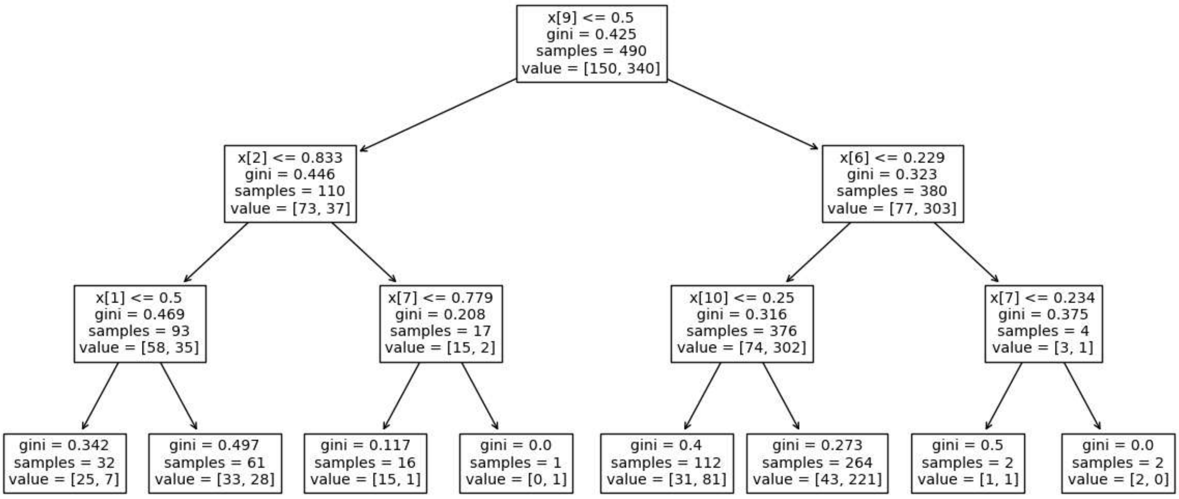
```
0.782258064516129
```

In [ ]:

```
# Plotting and visualizing the tree
```

In [291]:

```
plt.figure(figsize=(15,7))
plot_tree(dt)
plt.show()
```



Using the Random Forest method

In [295]:

```
from sklearn.ensemble import RandomForestClassifier
```

In [303]:

```
#To train a random forest, you need to specify the number of decision trees to use (the n_estimators parame

rf=RandomForestClassifier(n_estimators=50,max_depth=3,criterion='gini',random_state=20,oob_score=True)
```

In [304]:

```
rf.fit(X_train,y_train)
```

```
C:\Users\dtdee\AppData\Local\Temp\ipykernel_16536\1593328843.py:1: DataConversionWarning: A c
olumn-vector y was passed when a 1d array was expected. Please change the shape of y to (n_sa
mples,), for example using ravel().
  rf.fit(X_train,y_train)
```

Out[304]:

```
▾                       RandomForestClassifier
RandomForestClassifier(max_depth=3, n_estimators=50, oob_score=True,
                       random_state=20)
```

In [305]:

```
y_pred= rf.predict(X_test)
```

In [383]:

```
accuracy_rf=accuracy_score(y_pred,y_test)
accuracy_rf
```

Out[383]:

```
0.7741935483870968
```

In [312]:

```
# OOB Score is out of bag samples or samples taken other than training data which helps in model evaluation

rf.oob_score_
```

Out[312]:

```
0.7551020408163265
```

Using the SVM model

In [315]:

```
from sklearn.svm import SVC
```

In [367]:

```
# training=[]

# for i in range(10,20):
#     model=SVC(max_iter=i,random_state=10,kernel='rbf')
#     model.fit(X_train,y_train)
#     training.append(model)

# print(training)
```

In [370]:

```
svm=SVC(max_iter=10,random_state=10,kernel='rbf')
svm.fit(X_train,y_train)
```

```
C:\Users\dtdee\anaconda3\lib\site-packages\sklearn\utils\validation.py:1143: DataConversionWa
rning: A column-vector y was passed when a 1d array was expected. Please change the shape of
y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\dtdee\anaconda3\lib\site-packages\sklearn\svm\_base.py:299: ConvergenceWarning: Solv
er terminated early (max_iter=10).  Consider pre-processing your data with StandardScaler or
MinMaxScaler.
  warnings.warn(
```

Out[370]:

```
▾                SVC
SVC(max_iter=10, random_state=10)
```

In [371]:

```


y_pred=svm.predict(X_test)
```

In [372]:

```
accuracy_svm=accuracy_score(y_pred,y_test)
accuracy_svm
```

Out[372]:

```
0.7741935483870968
```

In [374]:

```
cross_val_score(svm,X_train,y_train,cv=5).mean()
```

```
C:\Users\dtdee\anaconda3\lib\site-packages\sklearn\utils\validation.py:1143: DataConversionWa
rning: A column-vector y was passed when a 1d array was expected. Please change the shape of
y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\dtdee\anaconda3\lib\site-packages\sklearn\svm\_base.py:299: ConvergenceWarning: Solv
er terminated early (max_iter=10).  Consider pre-processing your data with StandardScaler or
MinMaxScaler.
  warnings.warn(
C:\Users\dtdee\anaconda3\lib\site-packages\sklearn\utils\validation.py:1143: DataConversionWa
rning: A column-vector y was passed when a 1d array was expected. Please change the shape of
y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\dtdee\anaconda3\lib\site-packages\sklearn\svm\_base.py:299: ConvergenceWarning: Solv
er terminated early (max_iter=10).  Consider pre-processing your data with StandardScaler or
MinMaxScaler.
  warnings.warn(
C:\Users\dtdee\anaconda3\lib\site-packages\sklearn\utils\validation.py:1143: DataConversionWa
rning: A column-vector y was passed when a 1d array was expected. Please change the shape of
y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\dtdee\anaconda3\lib\site-packages\sklearn\svm\_base.py:299: ConvergenceWarning: Solv
er terminated early (max_iter=10).  Consider pre-processing your data with StandardScaler or
MinMaxScaler.
  warnings.warn(
C:\Users\dtdee\anaconda3\lib\site-packages\sklearn\utils\validation.py:1143: DataConversionWa
rning: A column-vector y was passed when a 1d array was expected. Please change the shape of
y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\dtdee\anaconda3\lib\site-packages\sklearn\svm\_base.py:299: ConvergenceWarning: Solv
er terminated early (max_iter=10).  Consider pre-processing your data with StandardScaler or
MinMaxScaler.
  warnings.warn(
C:\Users\dtdee\anaconda3\lib\site-packages\sklearn\utils\validation.py:1143: DataConversionWa
rning: A column-vector y was passed when a 1d array was expected. Please change the shape of
y to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
C:\Users\dtdee\anaconda3\lib\site-packages\sklearn\svm\_base.py:299: ConvergenceWarning: Solv
er terminated early (max_iter=10).  Consider pre-processing your data with StandardScaler or
MinMaxScaler.
  warnings.warn(
```

Out[374]:

```
0.7387755102040817
```

In [384]:

```
Best_model=pd.DataFrame({'Model':['Logistic Regression','Naive Bayes','Decision Tree','Random Forest','SVM'
                         'Accuracy_score':[Accuracy_LogReg,accuracy_GB,accuracy_dt,accuracy_rf,accuracy_svm
```

In [385]:

```
Best_model
```

Out[385]:

	Model	Accuracy_score
0	Logistic Regression	0.782258
1	Naive Bayes	0.774194
2	Decision Tree	0.782258
3	Random Forest	0.774194
4	SVM	0.774194

We see Decision Tree is giving bestaccuracy score so we can take the model .All other model are also good and they have accuracy very close by

In [272]:

```
# We can deploy this model now as we can check with a new customer coimng in for loan and with our model we

#Cust1  and cust2 comes in random for loan and Sales manager extracts all features as columns will be tsete


cust1= np.array([0,0 ,0 ,0,0 ,.221,0.022,0.2234,0.5412,0 ,1 ],ndmin=2)
cust1=cust1.reshape(1,11)
```

In [271]:

```
cust1.shape
```

Out[271]:

```
(1, 11)
```

In [391]:

```
# 1 shows --->Yes ,0 shows ----->No as the loan eligibilty
```

In [388]:

```
cust2= np.array([3,0 ,1 ,0, 0,.521,0.722,0.8234,0.2512,1 ,0 ],ndmin=2)
cust2=cust2.reshape(1,11)
```

In [389]:

```
cust2.shape
```

Out[389]:

```
(1, 11)
```

In [390]:

```
dt.predict(cust2)
```

Out[390]:

```
array([0.])
```

In [392]:

```
dt.predict(cust1)
```

Out[392]:

```
array([0.])
```

CONCLUSION

This is how we can easily make judgements of loan sanctioning with the best fit model

In [ ]:

```

```