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

%matplotlib inline
df=pd.read_csv('Admission_Predict.csv')
df
```

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	1	0.92
1	2	324	107	4	4.0	4.5	8.87	1	0.76
2	3	316	104	3	3.0	3.5	8.00	1	0.72
3	4	322	110	3	3.5	2.5	8.67	1	0.80
4	5	314	103	2	2.0	3.0	8.21	0	0.65
...	...	...	...	...	...	...	...	...	...
395	396	324	110	3	3.5	3.5	9.04	1	0.82
396	397	325	107	3	3.0	3.5	9.11	1	0.84
397	398	330	116	4	5.0	4.5	9.45	1	0.91
398	399	312	103	3	3.5	4.0	8.78	0	0.67
399	400	333	117	4	5.0	4.0	9.66	1	0.95

400 rows × 9 columns

```
df=df.drop(columns='Serial No.')
df
```

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	337	118	4	4.5	4.5	9.65	1	0.92
1	324	107	4	4.0	4.5	8.87	1	0.76
2	316	104	3	3.0	3.5	8.00	1	0.72
3	322	110	3	3.5	2.5	8.67	1	0.80
4	314	103	2	2.0	3.0	8.21	0	0.65
...	...	...	...	...	...	...	...	...
395	324	110	3	3.5	3.5	9.04	1	0.82
396	325	107	3	3.0	3.5	9.11	1	0.84
397	330	116	4	5.0	4.5	9.45	1	0.91
398	312	103	3	3.5	4.0	8.78	0	0.67
399	333	117	4	5.0	4.0	9.66	1	0.95

400 rows × 8 columns

```
ad=[]
for i in df['Chance of Admit']:
    if(i>=0.9):
        ad.append(1)
    else:
        ad.append(0)
```

```
df['Admission']=ad
df
```

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit	Admission
0	337	118	4	4.5	4.5	9.65	1	0.92	1
1	324	107	4	4.0	4.5	8.87	1	0.76	0
2	316	104	3	3.0	3.5	8.00	1	0.72	0
3	322	110	3	3.5	2.5	8.67	1	0.80	0
4	314	103	2	2.0	3.0	8.21	0	0.65	0
...	...	...	...	...	...	...	...	...	...
395	324	110	3	3.5	3.5	9.04	1	0.82	0
396	325	107	3	3.0	3.5	9.11	1	0.84	0
397	330	116	4	5.0	4.5	9.45	1	0.91	1
398	312	103	3	3.5	4.0	8.78	0	0.67	0
399	333	117	4	5.0	4.0	9.66	1	0.95	1

400 rows × 9 columns

```
df=df.drop(columns='Chance of Admit')
df
```

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Admission
0	337	118	4	4.5	4.5	9.65	1	1
1	324	107	4	4.0	4.5	8.87	1	0
2	316	104	3	3.0	3.5	8.00	1	0
3	322	110	3	3.5	2.5	8.67	1	0
4	314	103	2	2.0	3.0	8.21	0	0
...	...	...	...	...	...	...	...	...
395	324	110	3	3.5	3.5	9.04	1	0
396	325	107	3	3.0	3.5	9.11	1	0
397	330	116	4	5.0	4.5	9.45	1	1
398	312	103	3	3.5	4.0	8.78	0	0
399	333	117	4	5.0	4.0	9.66	1	1

400 rows × 8 columns

```
df.fillna(df.mean(),inplace=True)
df
```

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Admission
0	337	118	4	4.5	4.5	9.65	1	1

```
X=df[['GRE Score','TOEFL Score','University Rating','SOP','LOR','CGPA','Research']]
X
Y=df[['Admission']]
Y
```

	Admission
0	1
1	0
2	0
3	0
4	0
...	...
395	0
396	0
397	1
398	0
399	1

400 rows × 1 columns

```
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)
```

```
X_train
```

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research
328	324	112	4	4.0	3.5	8.77	1
380	322	104	3	3.5	4.0	8.84	1
203	334	120	5	4.0	5.0	9.87	1
280	311	102	3	4.5	4.0	8.64	1
372	336	119	4	4.5	4.0	9.62	1
...	...	...	...	...	...	...	...
70	332	118	5	5.0	5.0	9.64	1
76	327	112	3	3.0	3.0	8.72	1
118	296	99	2	3.0	3.5	7.28	0
278	308	103	2	3.0	3.5	8.49	0
345	316	98	1	1.5	2.0	7.43	0

320 rows × 7 columns

```
X_test
```

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research
112	301	107	3	3.5	3.5	8.34	1
151	332	116	5	5.0	5.0	9.28	1
304	313	106	2	2.5	2.0	8.43	0
28	295	93	1	2.0	2.0	7.20	0
275	322	110	3	3.5	3.0	8.96	1
...	...	...	...	...	...	...	...
135	314	109	4	3.5	4.0	8.77	1
101	312	105	2	2.5	3.0	8.12	0

```
Y_train
```

	Admission
328	0
380	0
203	1
280	0
372	1
...	...
70	1
76	0
118	0
278	0
345	0

320 rows × 1 columns

```
Y_test
```

	Admission
112	0
151	1
304	0
28	0
275	0
...	...
135	0
101	0
326	0
66	0
25	1

80 rows × 1 columns

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn import tree
DT1= tree.DecisionTreeClassifier(criterion = "gini",random_state = 100)
DT1=DT1.fit(X_train,Y_train)
y_predict = DT1.predict(X_test)
```

```
y_predict
```

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

```
print("Testing accuracy is",accuracy_score(Y_test,y_predict))
```

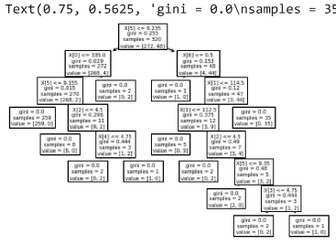
```
Testing accuracy is 0.95
```

```
y_predict = DT1.predict(X_train)
print("Training accuracy is",accuracy_score(y_predict,Y_train))
```

```
Training accuracy is 1.0
```

```
tree.plot_tree(DT1)
```

```
[Text(0.4166666666666667, 0.9375, 'X[5] <= 9.235\ngini = 0.255\nsamples = 320\nvalue = [272, 48]'),
 Text(0.25, 0.8125, 'X[0] <= 335.0\ngini = 0.029\nsamples = 272\nvalue = [268, 4]'),
 Text(0.16666666666666666, 0.6875, 'X[5] <= 9.155\ngini = 0.015\nsamples = 270\nvalue = [268, 2]'),
 Text(0.08333333333333333, 0.5625, 'gini = 0.0\nsamples = 259\nvalue = [259, 0]'),
 Text(0.25, 0.5625, 'X[2] <= 4.5\ngini = 0.298\nsamples = 11\nvalue = [9, 2]'),
 Text(0.16666666666666666, 0.4375, 'gini = 0.0\nsamples = 8\nvalue = [8, 0]'),
 Text(0.3333333333333333, 0.4375, 'X[4] <= 4.75\ngini = 0.444\nsamples = 3\nvalue = [1, 2]'),
 Text(0.25, 0.3125, 'gini = 0.0\nsamples = 2\nvalue = [0, 2]'),
 Text(0.4166666666666667, 0.3125, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(0.3333333333333333, 0.6875, 'gini = 0.0\nsamples = 2\nvalue = [0, 2]'),
 Text(0.5833333333333334, 0.8125, 'X[6] <= 0.5\ngini = 0.153\nsamples = 48\nvalue = [4, 44]'),
 Text(0.5, 0.6875, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(0.6666666666666666, 0.6875, 'X[1] <= 114.5\ngini = 0.12\nsamples = 47\nvalue = [3, 44]'),
 Text(0.5833333333333334, 0.5625, 'X[1] <= 112.5\ngini = 0.375\nsamples = 12\nvalue = [3, 9]'),
 Text(0.5, 0.4375, 'gini = 0.0\nsamples = 5\nvalue = [0, 5]'),
 Text(0.6666666666666666, 0.4375, 'X[2] <= 4.5\ngini = 0.49\nsamples = 7\nvalue = [3, 4]'),
 Text(0.5833333333333334, 0.3125, 'gini = 0.0\nsamples = 2\nvalue = [0, 2]'),
 Text(0.75, 0.3125, 'X[5] <= 9.35\ngini = 0.48\nsamples = 5\nvalue = [3, 2]'),
 Text(0.6666666666666666, 0.1875, 'gini = 0.0\nsamples = 2\nvalue = [2, 0]'),
 Text(0.8333333333333334, 0.1875, 'X[3] <= 4.75\ngini = 0.444\nsamples = 3\nvalue = [1, 2]'),
 Text(0.75, 0.0625, 'gini = 0.0\nsamples = 2\nvalue = [0, 2]'),
 Text(0.9166666666666666, 0.0625, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(0.75, 0.5625, 'gini = 0.0\nsamples = 35\nvalue = [0, 35]')]
```



```
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(DT1,X_test,Y_test)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function plot_confusion_matrix is deprecated; Function `plot_confusion_matrix` is depre
  warnings.warn(msg, category=FutureWarning)
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fc6ee9d6290>
```

```
predicted= pd.DataFrame(y_predict)
predicted
```

	0
0	0
1	0
2	1
3	0
4	1
...	...
315	1
316	0
317	0
318	0
319	0

320 rows × 1 columns

```
DT2= tree.DecisionTreeClassifier(criterion = "entropy",random_state = 100)
DT2=DT2.fit(X_train,Y_train)
y_predict = DT2.predict(X_test)
y_predict
```

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

```
print("Testing accuracy is",accuracy_score(Y_test,y_predict))
```

```
Testing accuracy is 0.975
```

```
y_predict = DT1.predict(X_train)
print("Training accuracy is",accuracy_score(y_predict,Y_train))
```

```
Training accuracy is 1.0
```

```
tree.plot_tree(DT2)
```

```
[Text(0.4117647058823529, 0.9285714285714286, 'X[5] <= 9.185\nentropy = 0.61\nsamples = 320\nvalue = [272, 48]'),
 Text(0.11764705882352941, 0.7857142857142857, 'X[5] <= 9.155\nentropy = 0.036\nsamples = 264\nvalue = [263, 1]'),
 Text(0.058823529411764705, 0.6428571428571429, 'entropy = 0.0\nsamples = 259\nvalue = [259, 0]'),
 Text(0.17647058823529413, 0.6428571428571429, 'X[2] <= 4.5\nentropy = 0.722\nsamples = 5\nvalue = [4, 1]'),
 Text(0.11764705882352941, 0.5, 'entropy = 0.0\nsamples = 4\nvalue = [4, 0]'),
 Text(0.23529411764705882, 0.5, 'entropy = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(0.7058823529411765, 0.7857142857142857, 'X[1] <= 114.5\nentropy = 0.636\nsamples = 56\nvalue = [9, 47]'),
 Text(0.5294117647058824, 0.6428571428571429, 'X[5] <= 9.35\nentropy = 0.991\nsamples = 18\nvalue = [8, 10]'),
 Text(0.35294117647058826, 0.5, 'X[3] <= 4.25\nentropy = 0.881\nsamples = 10\nvalue = [7, 3]'),
 Text(0.29411764705882354, 0.35714285714285715, 'entropy = 0.0\nsamples = 5\nvalue = [5, 0]'),
 Text(0.4117647058823529, 0.35714285714285715, 'X[0] <= 325.5\nentropy = 0.971\nsamples = 5\nvalue = [2, 3]'),
 Text(0.35294117647058826, 0.21428571428571427, 'entropy = 0.0\nsamples = 2\nvalue = [0, 2]'),
 Text(0.47058823529411764, 0.21428571428571427, 'X[5] <= 9.235\nentropy = 0.918\nsamples = 3\nvalue = [2, 1]'),
 Text(0.4117647058823529, 0.07142857142857142, 'entropy = 0.0\nsamples = 2\nvalue = [2, 0]'),
 Text(0.5294117647058824, 0.07142857142857142, 'entropy = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(0.7058823529411765, 0.5, 'X[0] <= 330.0\nentropy = 0.544\nsamples = 8\nvalue = [1, 7]'),
 Text(0.6470588235294118, 0.35714285714285715, 'X[0] <= 327.5\nentropy = 0.918\nsamples = 3\nvalue = [1, 2]'),
```

```
plot_confusion_matrix(DT2,X_test,Y_test)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function plot_confusion_matrix is deprecated; Function `plot_confusion_matrix` is depre
  warnings.warn(msg, category=FutureWarning)
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7fc6ee4b5710>
```


```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
y_predict = DT2.predict(X_test)
print("Confusion Matrix: ",
        confusion_matrix(Y_test, y_predict))

print("Accuracy : ",
    accuracy_score(Y_test,y_predict))

print("Report : ",
    classification_report(Y_test, y_predict))
```

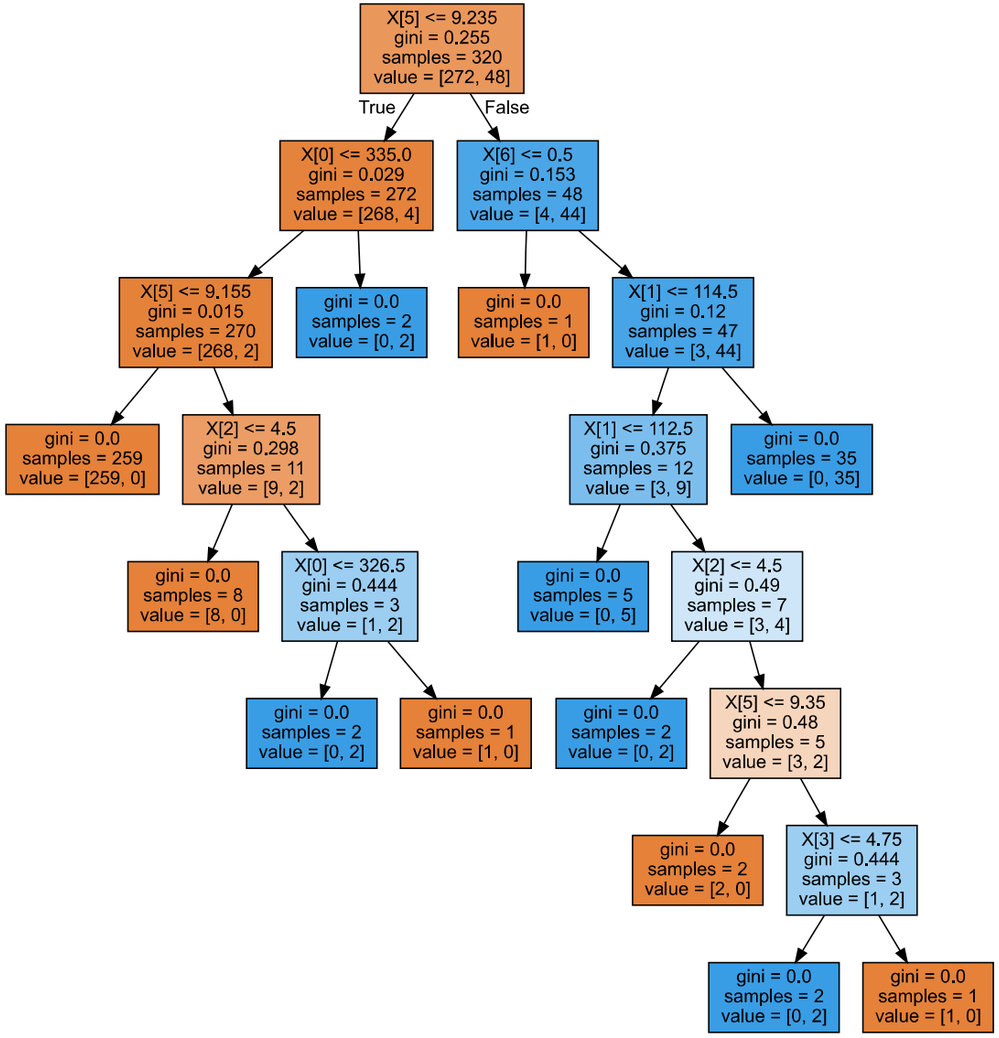
```
Confusion Matrix:  [[70  1]
 [ 1  8]]
Accuracy :  0.975
Report :                precision    recall  f1-score   support

           0       0.99      0.99      0.99        71
           1       0.89      0.89      0.89         9

    accuracy                           0.97        80
   macro avg       0.94      0.94      0.94        80
weighted avg       0.97      0.97      0.97        80
```

```
import graphviz
from sklearn import tree
clf=tree.DecisionTreeClassifier()
clf=clf.fit(X_train,Y_train)

dot_data=tree.export_graphviz(clf,filled=True)
graph=graphviz.Source(dot_data)
graph
```


X[5] <= 9.235
gini = 0.255
samples = 320
value = [272, 48]
True
False
X[0] <= 335.0
gini = 0.029
samples = 272
value = [268, 4]
X[6] <= 0.5
gini = 0.153
samples = 48
value = [4, 44]
X[5] <= 9.155
gini = 0.015
samples = 270
value = [268, 2]
gini = 0.0
samples = 2
value = [0, 2]
gini = 0.0
samples = 1
value = [1, 0]
X[1] <= 114.5
gini = 0.12
samples = 47
value = [3, 44]
gini = 0.0
samples = 259
value = [259, 0]
X[2] <= 4.5
gini = 0.298
samples = 11
value = [9, 2]
X[1] <= 112.5
gini = 0.375
samples = 12
value = [3, 9]
gini = 0.0
samples = 35
value = [0, 35]
gini = 0.0
samples = 8
value = [8, 0]
X[0] <= 326.5
gini = 0.444
samples = 3
value = [1, 2]
gini = 0.0
samples = 5
value = [0, 5]
X[2] <= 4.5
gini = 0.49
samples = 7
value = [3, 4]
gini = 0.0
samples = 2
value = [0, 2]
gini = 0.0
samples = 1
value = [1, 0]
gini = 0.0
samples = 2
value = [0, 2]
X[5] <= 9.35
gini = 0.48
samples = 5
value = [3, 2]
gini = 0.0
samples = 2
value = [2, 0]
X[3] <= 4.75
gini = 0.444
samples = 3
value = [1, 2]
gini = 0.0
samples = 2
value = [0, 2]
gini = 0.0
samples = 1
value = [1, 0]


Colab paid products - Cancel contracts here
0s completed at 1:57 PM