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


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

New Section

```
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

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

```
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
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

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

Y=df[['Admission']]
X,Y
```

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

 [400 rows x 7 columns],      Admission
0           1
1           0
2           0
3           0
4           0
..        ...
395         0
396         0
397         1
398         0
399         1

[400 rows x 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
303	323	107	3	3.5	3.5	8.55	1
328	324	112	4	4.0	3.5	8.77	1
22	328	116	5	5.0	5.0	9.50	1
192	322	114	5	4.5	4.0	8.94	1
321	323	104	3	4.0	4.0	8.44	1
...	...	...	...	...	...	...	...
45	322	110	5	5.0	4.0	9.10	1
103	317	104	2	4.5	4.0	8.47	0
121	334	119	5	4.5	4.5	9.48	1
236	325	112	4	4.0	4.5	9.17	1
207	310	102	3	3.5	4.0	8.02	1

320 rows × 7 columns

```
X_test
```

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research
16	317	107	3	4.0	3.0	8.70	0
186	317	107	3	3.5	3.0	8.68	1
216	322	112	4	4.5	4.5	9.26	1
334	312	107	4	4.5	4.0	8.65	1
254	321	114	4	4.0	5.0	9.12	0
...	...	...	...	...	...	...	...
4	314	103	2	2.0	3.0	8.21	0
237	329	114	5	4.5	5.0	9.19	1
305	321	109	3	3.5	3.5	8.80	1
47	339	119	5	4.5	4.0	9.70	0
263	324	111	3	2.5	1.5	8.79	1

80 rows × 7 columns

```
Y_train
```

```
     Admission
303          0
328          0
 22          1
```

```
Y_test
```

```
     Admission
 16          0
186          0
216          1
334          0
254          0
...        ...
  4          0
237          0
305          0
 47          0
263          0
80 rows × 1 columns
```

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

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

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

```
Testing accuracy is 0.9
```

```
y_predict = DT1.predict(X_train)
```

```
print("Training accuracy is",accuracy_score(Y_train,y_predict))
```

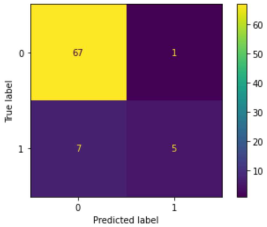
```
Training accuracy is 1.0
```

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

```
[Text(0.5, 0.9166666666666666, 'X[5] <= 9.235\nentropy = 0.586\nsamples = 320\nvalue = [275, 45]'),
 Text(0.2, 0.75, 'X[1] <= 118.0\nentropy = 0.035\nsamples = 272\nvalue = [271, 1]')
```

```
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 deprecat
  warnings.warn(msg, category=FutureWarning)
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f46214690d0>
```



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

```
       0
  0    0
  1    0
  2    1
  3    0
  4    0
 ...  ...
315    0
316    0
317    1
318    0
319    0
320 rows × 1 columns
```

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

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

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

```
Testing accuracy is 0.9
```

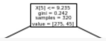
```
y_predict = DT2.predict(X_train)
```

```
print("Training accuracy is",accuracy_score(Y_train,y_predict))
```

```
Training accuracy is 1.0
```

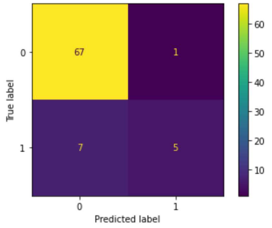
```
tree.plot_tree(DT2)
```

```
[Text(0.4230769230769231, 0.9, 'X[5] <= 9.235\ngini = 0.242\nsamples = 320\nvalue = [275, 45]'),
 Text(0.15384615384615385, 0.7, 'X[1] <= 118.0\ngini = 0.007\nsamples = 272\nvalue = [271, 1]'),
 Text(0.07692307692307693, 0.5, 'gini = 0.0\nsamples = 270\nvalue = [270, 0]'),
 Text(0.23076923076923078, 0.5, 'X[3] <= 4.25\ngini = 0.5\nsamples = 2\nvalue = [1, 1]'),
 Text(0.15384615384615385, 0.3, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(0.3076923076923077, 0.3, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(0.6923076923076923, 0.7, 'X[0] <= 329.5\ngini = 0.153\nsamples = 48\nvalue = [4, 44]'),
 Text(0.5384615384615384, 0.5, 'X[1] <= 114.5\ngini = 0.48\nsamples = 5\nvalue = [3, 2]'),
 Text(0.46153846153846156, 0.3, 'gini = 0.0\nsamples = 3\nvalue = [3, 0]'),
 Text(0.6153846153846154, 0.3, 'gini = 0.0\nsamples = 2\nvalue = [0, 2]'),
 Text(0.8461538461538461, 0.5, 'X[5] <= 9.295\ngini = 0.045\nsamples = 43\nvalue = [1, 42]'),
 Text(0.7692307692307693, 0.3, 'X[0] <= 332.5\ngini = 0.32\nsamples = 5\nvalue = [1, 4]'),
 Text(0.6923076923076923, 0.1, 'gini = 0.0\nsamples = 4\nvalue = [0, 4]'),
 Text(0.8461538461538461, 0.1, 'gini = 0.0\nsamples = 1\nvalue = [1, 0]'),
 Text(0.9230769230769231, 0.3, 'gini = 0.0\nsamples = 38\nvalue = [0, 38]')]
```



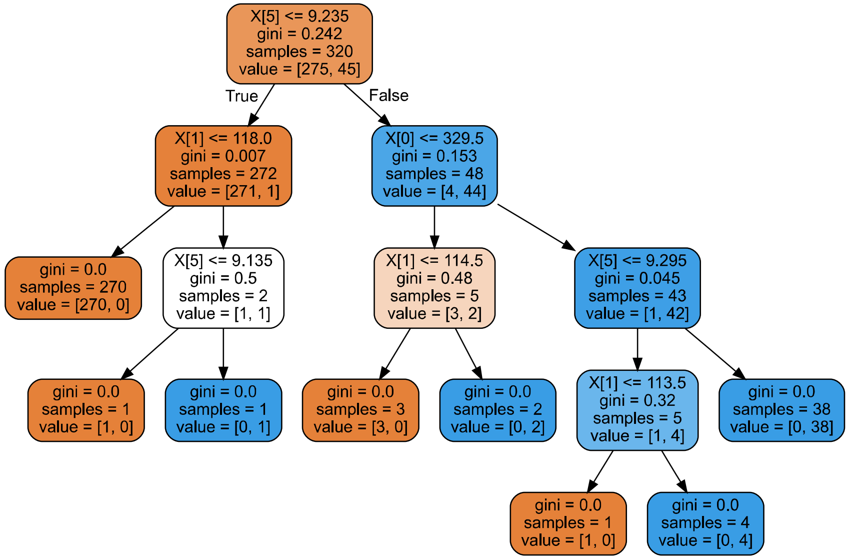
```
plot_confusion_matrix(DT2,X_test,Y_test)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: Function plot_confusion_matrix is deprecated; Function `plot_confusion_matrix` is deprecat
  warnings.warn(msg, category=FutureWarning)
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f4620e6fd50>
```



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

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



✓ 0s completed at 10:08 PM ● ×