In [1]:
```
from sklearn.model_selection import train_test_split
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import f1_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
import matplotlib.pyplot as plt
```

In [4]:
```
df=pd.read_csv("/sameer/Admission_Predict.csv - Admission_Predict.csv")
```

In [5]:
```
df
```

Out[5]:

	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

In [6]:
```
df.shape
```

Out[6]:
```
(400, 9)
```

In [7]:
```
df.columns
```

Out[7]:
```
Index(['Serial No.', 'GRE Score', 'TOEFL Score', 'University Rating', 'SOP',
       'LOR', 'CGPA', 'Research', 'Chance of Admit'],
      dtype='object')
```

In [8]:
```
df.dropna(inplace=True)
```

In [9]:
```
df.shape
```

Out[9]:
```
(400, 9)
```

In [10]:
```
print(df.describe())
```

```
       Serial No.   GRE Score  TOEFL Score  University Rating         SOP  \
count  400.000000  400.000000   400.000000         400.000000  400.000000   
mean   200.500000  316.807500   107.410000           3.087500    3.400000   
std    115.614301   11.473646     6.069514           1.143728    1.006869   
min      1.000000  290.000000    92.000000           1.000000    1.000000   
25%    100.750000  308.000000   103.000000           2.000000    2.500000   
50%    200.500000  317.000000   107.000000           3.000000    3.500000   
75%    300.250000  325.000000   112.000000           4.000000    4.000000   
max    400.000000  340.000000   120.000000           5.000000    5.000000   

              LOR        CGPA    Research  Chance of Admit  
count  400.000000  400.000000  400.000000       400.000000  
mean     3.452500    8.598925    0.547500         0.724350  
std      0.898478    0.596317    0.498362         0.142609  
min      1.000000    6.800000    0.000000         0.340000  
25%      3.000000    8.170000    0.000000         0.640000  
50%      3.500000    8.610000    1.000000         0.730000  
75%      4.000000    9.062500    1.000000         0.830000  
max      5.000000    9.920000    1.000000         0.970000  
```

In [11]:
```
print(df.head(10))
```

```
   Serial No.  GRE Score  TOEFL Score  University Rating  SOP  LOR  CGPA  \
0           1        337          118                  4  4.5  4.5  9.65   
1           2        324          107                  4  4.0  4.5  8.87   
2           3        316          104                  3  3.0  3.5  8.00   
3           4        322          110                  3  3.5  2.5  8.67   
4           5        314          103                  2  2.0  3.0  8.21   
5           6        330          115                  5  4.5  3.0  9.34   
6           7        321          109                  3  3.0  4.0  8.20   
7           8        308          101                  2  3.0  4.0  7.90   
8           9        302          102                  1  2.0  1.5  8.00   
9          10        323          108                  3  3.5  3.0  8.60   

   Research  Chance of Admit  
0         1             0.92  
1         1             0.76  
2         1             0.72  
3         1             0.80  
4         0             0.65  
5         1             0.90  
6         1             0.75  
7         0             0.68  
8         0             0.50  
9         0             0.45  
```

In [13]:

```
X = df[['GRE Score', 'TOEFL Score', 'CGPA']]
Y = df['Chance of Admit']
```

In [14]:

```
X.shape
```

Out[14]:

```
(400, 3)
```

In [16]:

```
X
```

Out[16]:

	GRE Score	TOEFL Score	CGPA
0	337	118	9.65
1	324	107	8.87
2	316	104	8.00
3	322	110	8.67
4	314	103	8.21
...	...	...	...
395	324	110	9.04
396	325	107	9.11
397	330	116	9.45
398	312	103	8.78
399	333	117	9.66

400 rows × 3 columns

In [17]:

```
Y.shape
```

Out[17]:

```
(400,)
```

In [18]:

```
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2, random_state=1)
```

In [19]:

```
y_train.shape
```

Out[19]:

```
(320,)
```

In [20]:

```
print(X_train.head())
```

```
     GRE Score  TOEFL Score  CGPA
93         301           97  7.88
23         334          119  9.70
299        305          112  8.65
13         307          109  8.00
90         318          106  7.92
```

```
In [21]: # chose 0.82 because it is the 3rd quartile for chance of admit
         ty_train=[1 if Y > 0.82 else 0 for Y  in y_train]
         ty_train=np.array(ty_train)
```

```
In [22]: ty_train
```

```
Out[22]: array([0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0,
                0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
                0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1,
                0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
                0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0,
                0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0,
                0, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
                0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0,
                0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1,
                0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0,
                0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1,
                1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0])
```

```
In [23]: ty_train.shape
```

```
Out[23]: (320,)
```

```
In [24]: y_test.shape
```

```
Out[24]: (80,)
```

```
In [25]: ty_test=[1 if Y > 0.82 else 0 for Y  in y_test]
         ty_test=np.array(ty_test)
```

```
In [26]: ty_test.shape
```

```
Out[26]: (80,)
```

```
In [27]: tree_model = DecisionTreeClassifier(criterion='gini', max_depth=3, random_state=1)
```

```
In [28]: tree_model.fit(X_train, ty_train)
```

```
Out[28]: DecisionTreeClassifier(max_depth=3, random_state=1)
```

```
In [29]: dt_pred = tree_model.predict(X_test)
```

In [30]:

```
print("Actual - Predicted \n")
for x in ty_test:
    for y in dt_pred:
        print(x,"         ",y)
```

```
Actual - Predicted

0           0
0           0
0           0
0           0
0           1
0           0
0           0
0           0
0           0
0           1
0           0
0           1
0           1
0           0
0           0
0           0
0           1
0           1
0           0
0           0
0           0
0           0
0           1
0           1
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           1
0           0
0           1
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
             0         1
             0         0
             0         0
             0         0
             0         0
             0         1
             0         0
             0         1
             0         1
             0         0
             0         0
             0         0
             0         1
             0         1
             0         0
             0         0
             0         0
             0         0
             0         1
             0         1
             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         1
             0         0
             0         1
             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           1
0           0
0           0
0           0
0           0
0           1
0           0
0           1
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
```

In [31]:
```
from sklearn import metrics
```

In [32]:
```
print(metrics.confusion_matrix(ty_test, dt_pred))
```

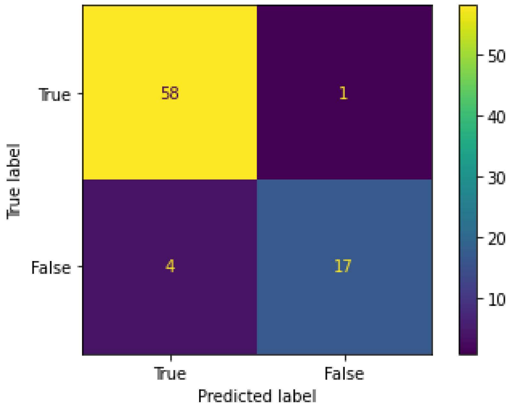
```
[[58  1]
 [ 4 17]]
```

In [33]:

```
confusion_matrix = metrics.confusion_matrix(ty_test, dt_pred)

cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = confusion_matrix, display_labels = [True, False])

cm_display.plot()
plt.show()
```



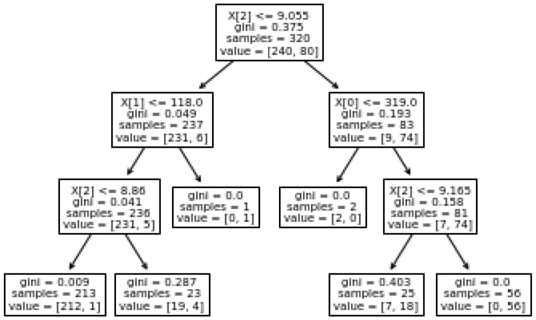
In [34]:

```
print("Decision Tree Accuracy: %.3f" % accuracy_score(ty_test, dt_pred))
print("Decision Tree F1-Score: %.3f" % f1_score(ty_test, dt_pred))
print("Decision Tree Precision: %.3f" % precision_score(ty_test, dt_pred))
print("Decision Tree Recall: %.3f" % recall_score(ty_test, dt_pred))
```

```
Decision Tree Accuracy: 0.938
Decision Tree F1-Score: 0.872
Decision Tree Precision: 0.944
Decision Tree Recall: 0.810
```

In [35]:
```
from sklearn import tree
tree.plot_tree(tree_model)
```

Out[35]:
```
[Text(0.5, 0.875, 'X[2] <= 9.055\ngini = 0.375\nsamples = 320\nvalue = [240, 80]'),
 Text(0.3, 0.625, 'X[1] <= 118.0\ngini = 0.049\nsamples = 237\nvalue = [231, 6]'),
 Text(0.2, 0.375, 'X[2] <= 8.86\ngini = 0.041\nsamples = 236\nvalue = [231, 5]'),
 Text(0.1, 0.125, 'gini = 0.009\nsamples = 213\nvalue = [212, 1]'),
 Text(0.3, 0.125, 'gini = 0.287\nsamples = 23\nvalue = [19, 4]'),
 Text(0.4, 0.375, 'gini = 0.0\nsamples = 1\nvalue = [0, 1]'),
 Text(0.7, 0.625, 'X[0] <= 319.0\ngini = 0.193\nsamples = 83\nvalue = [9, 74]'),
 Text(0.6, 0.375, 'gini = 0.0\nsamples = 2\nvalue = [2, 0]'),
 Text(0.8, 0.375, 'X[2] <= 9.165\ngini = 0.158\nsamples = 81\nvalue = [7, 74]'),
 Text(0.7, 0.125, 'gini = 0.403\nsamples = 25\nvalue = [7, 18]'),
 Text(0.9, 0.125, 'gini = 0.0\nsamples = 56\nvalue = [0, 56]')]
```



In [36]:
```
new_data=[[337,118,9.88]]
```

In [37]:
```
prediction = tree_model.predict(new_data)
```

```
C:\Users\samir\Anaconda3\lib\site-packages\sklearn\base.py:451: UserWarning: X does not have valid feature names, but DecisionTreeClassifier
was fitted with feature names
  "X does not have valid feature names, but"
```

In [38]:
```
if prediction == 1:
    print("Student get admission" )
else:
     print("Student not get admission" )
```

```
Student get admission
```

In [ ]: