In [1]:

```
import pandas as pd
import seaborn as sns
```

In [2]:

```
df = pd.read_csv('Admission_Predict.csv')
df.columns
```

Out[2]:

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

In [3]:

```
df.head()
```

Out[3]:

	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

In [5]:

```
from sklearn.preprocessing import Binarizer
bi = Binarizer(threshold = 0.75)
df['Chance of Admit '] = bi.fit_transform(df[['Chance of Admit ']])
```

In [6]:

```
df.head()
```

Out[6]:

	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	1.0
1	2	324	107	4	4.0	4.5	8.87	1	1.0
2	3	316	104	3	3.0	3.5	8.00	1	0.0
3	4	322	110	3	3.5	2.5	8.67	1	1.0
4	5	314	103	2	2.0	3.0	8.21	0	0.0

In [7]:

```
x = df.drop('Chance of Admit ', axis = 1)
y = df['Chance of Admit ']
x
```

Out[7]:

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

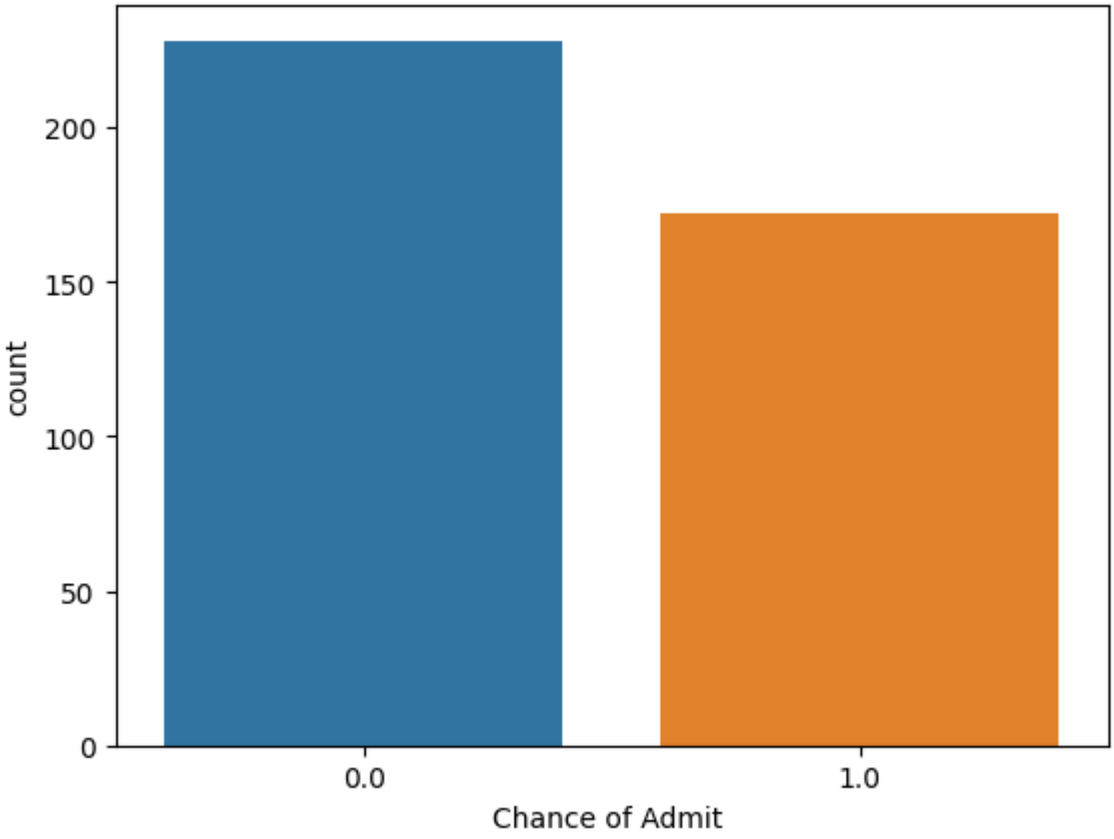
400 rows × 8 columns

In [9]:
```
y.astype('int')
```

Out[9]:
```
0      1
1      1
2      0
3      1
4      0
      ..
395    1
396    1
397    1
398    0
399    1
Name: Chance of Admit , Length: 400, dtype: int64
```

In [10]:
```
sns.countplot(x = y)
```

Out[10]:
```
<AxesSubplot: xlabel='Chance of Admit ', ylabel='count'>
```



```
In [11]: y.value_counts()

Out[11]: 0.0    228
         1.0    172
         Name: Chance of Admit , dtype: int64

In [12]: # cross validation
         from sklearn.model_selection import train_test_split
         x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=0, test_s

In [13]: x_train.shape

Out[13]: (300, 8)

In [14]: x_test.shape

Out[14]: (100, 8)

In [15]: x_test
```

Out[15]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research
132	133	309	105	5	3.5	3.5	8.56	0
309	310	308	110	4	3.5	3.0	8.60	0
341	342	326	110	3	3.5	3.5	8.76	1
196	197	306	105	2	3.0	2.5	8.26	0
246	247	316	105	3	3.0	3.5	8.73	0
...	...	...	...	...	...	...	...	...
146	147	315	105	3	2.0	2.5	8.48	0
135	136	314	109	4	3.5	4.0	8.77	1
390	391	314	102	2	2.0	2.5	8.24	0
264	265	325	110	2	3.0	2.5	8.76	1
364	365	313	102	3	3.5	4.0	8.90	1

100 rows × 8 columns

In [16]:

```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier (random_state = 0)
```

In [17]:

```
classifier.fit(x_train, y_train)
```

Out[17]:

```
▾        DecisionTreeClassifier

DecisionTreeClassifier(random_state=0)
```

In [18]:

```
y_pred = classifier.predict(x_test)
```

In [19]:

```
result = pd.DataFrame({
    'actual': y_test,
    'predicted' : y_pred
})
result
```

Out[19]:

	actual	predicted
132	0.0	0.0
309	0.0	0.0
341	1.0	1.0
196	0.0	0.0
246	0.0	1.0
...	...	...
146	0.0	0.0
135	1.0	1.0
390	0.0	0.0
264	0.0	0.0
364	1.0	1.0

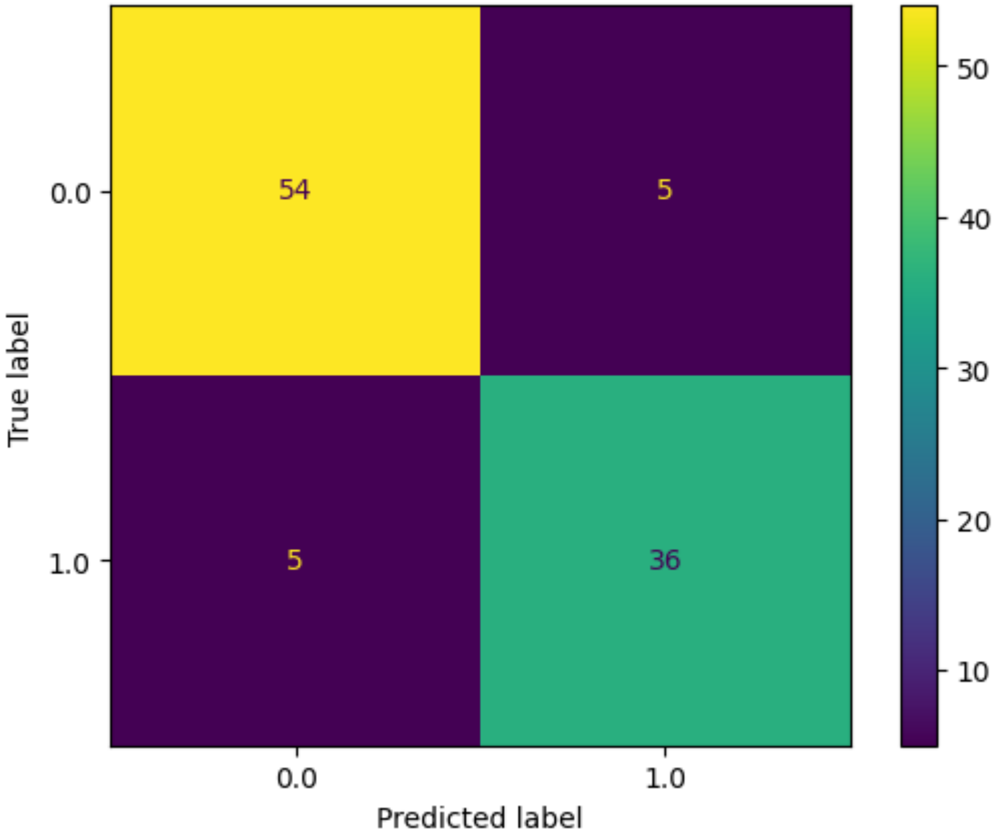
100 rows × 2 columns

In [21]:

```
from sklearn.metrics import ConfusionMatrixDisplay, accuracy_score
from sklearn.metrics import classification_report
ConfusionMatrixDisplay.from_predictions(y_test, y_pred)
```

Out[21]:

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f95cf01022
0>
```



In [22]:

```
accuracy_score(y_test, y_pred)
```

Out[22]:

```
0.9
```

In [23]:
```
print(classification_report(y_test, y_pred))
```

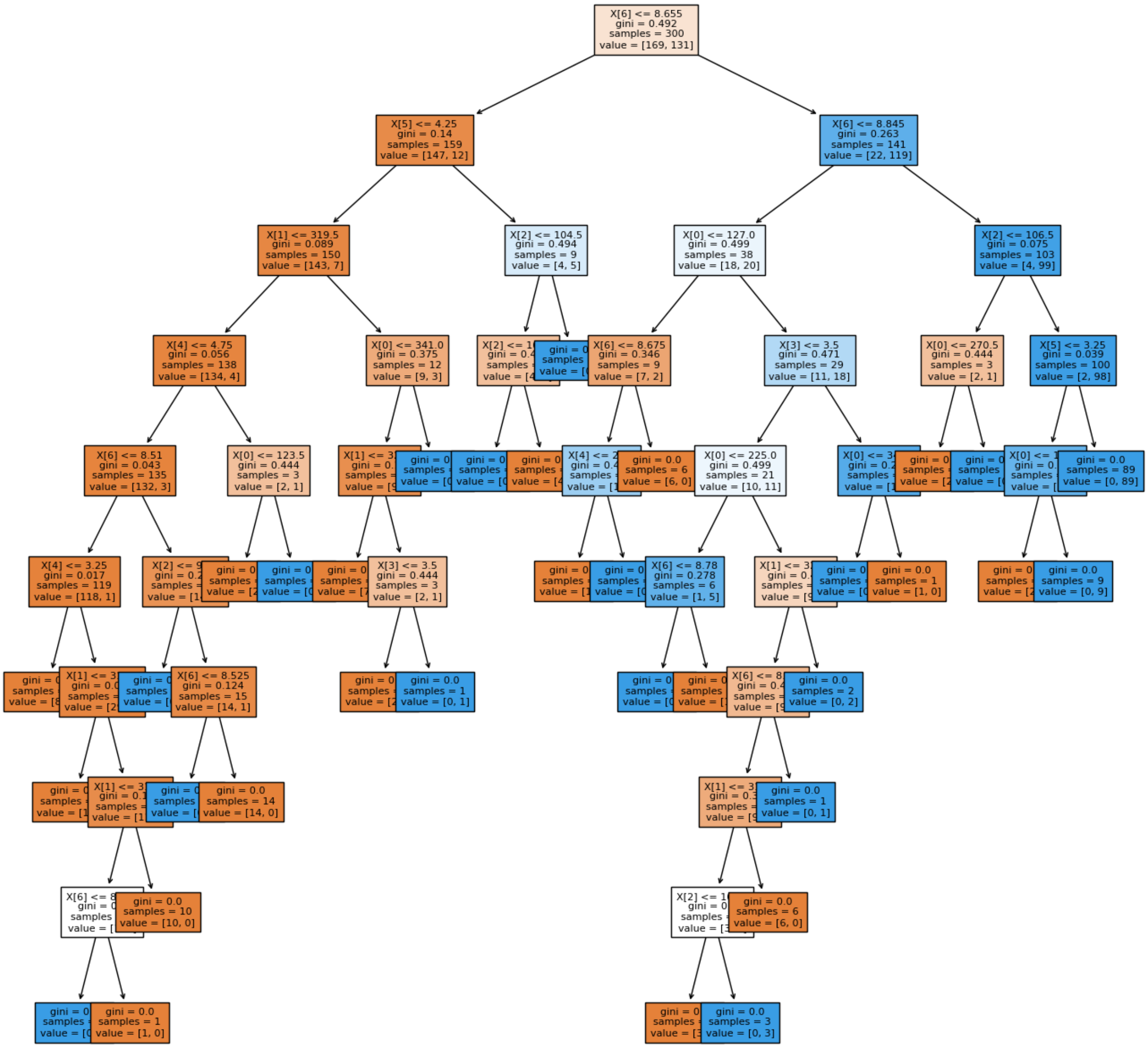
```
              precision    recall  f1-score   support

         0.0       0.92      0.92      0.92        59
         1.0       0.88      0.88      0.88        41

    accuracy                           0.90       100
   macro avg       0.90      0.90      0.90       100
weighted avg       0.90      0.90      0.90       100
```

In [24]:
```
new = [[136, 314, 109, 4, 3.5, 4.0, 8.77, 1]]
classifier.predict(new)[0]
```

```
/home/sapatevaibhav/.local/lib/python3.10/site-packages/sklearn/base.py:450: User
Warning: X does not have valid feature names, but DecisionTreeClassifier was fitt
ed with feature names
  warnings.warn(
```

Out[24]:
```
1.0
```

In [30]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(16,16))
from sklearn.tree import plot_tree
plot_tree(classifier, fontsize = 8, filled = True);
```



X[6] <= 8.655 gini = 0.492 samples = 300 value = [169, 131]
X[5] <= 4.25 gini = 0.14 samples = 159 value = [147, 12]
X[6] <= 8.845 gini = 0.263 samples = 141 value = [22, 119]
X[1] <= 319.5 gini = 0.089 samples = 150 value = [143, 7]
X[2] <= 104.5 gini = 0.494 samples = 9 value = [4, 5]
X[0] <= 127.0 gini = 0.499 samples = 38 value = [18, 20]
X[2] <= 106.5 gini = 0.075 samples = 103 value = [4, 99]
X[4] <= 4.75 gini = 0.056 samples = 138 value = [134, 4]
X[0] <= 341.0 gini = 0.375 samples = 12 value = [9, 3]
X[6] <= 8.675 gini = 0.346 samples = 9 value = [7, 2]
X[3] <= 3.5 gini = 0.471 samples = 29 value = [11, 18]
X[0] <= 270.5 gini = 0.444 samples = 3 value = [2, 1]
X[5] <= 3.25 gini = 0.039 samples = 100 value = [2, 98]
X[6] <= 8.51 gini = 0.043 samples = 135 value = [132, 3]
X[0] <= 123.5 gini = 0.444 samples = 3 value = [2, 1]
gini = 0.0 samples = 6 value = [6, 0]
X[0] <= 225.0 gini = 0.499 samples = 21 value = [10, 11]
gini = 0.0 samples = 89 value = [0, 89]
X[4] <= 3.25 gini = 0.017 samples = 119 value = [118, 1]
X[3] <= 3.5 gini = 0.444 samples = 3 value = [2, 1]
X[6] <= 8.78 gini = 0.278 samples = 6 value = [1, 5]
gini = 0.0 samples = 1 value = [1, 0]
gini = 0.0 samples = 9 value = [0, 9]
X[6] <= 8.525 gini = 0.124 samples = 15 value = [14, 1]
gini = 0.0 samples = 1 value = [0, 1]
gini = 0.0 samples = 2 value = [0, 2]
gini = 0.0 samples = 14 value = [14, 0]
gini = 0.0 samples = 1 value = [0, 1]
gini = 0.0 samples = 10 value = [10, 0]
gini = 0.0 samples = 6 value = [6, 0]
gini = 0.0 samples = 1 value = [1, 0]
gini = 0.0 samples = 3 value = [0, 3]

In [ ]: