In [1]:

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

In [7]:

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

In [8]:

```
df.columns
```

Out[8]:

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

In [9]:

```
df.shape
```

Out[9]:

```
(400, 9)
```

In [10]:

```
df.head()
```

Out[10]:

	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 [13]:

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

In [14]:

```
df.head()
```

Out[14]:

	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 [15]:

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

In [16]:

```
x
```

Out[16]:

	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 [17]:

```
y
```

Out[17]:

```
0      1.0
1      1.0
2      0.0
3      1.0
4      0.0
      ...
395    1.0
396    1.0
397    1.0
398    0.0
399    1.0
Name: Chance of Admit , Length: 400, dtype: float64
```

In [19]:

```
y=y.astype('int')
```

In [20]:

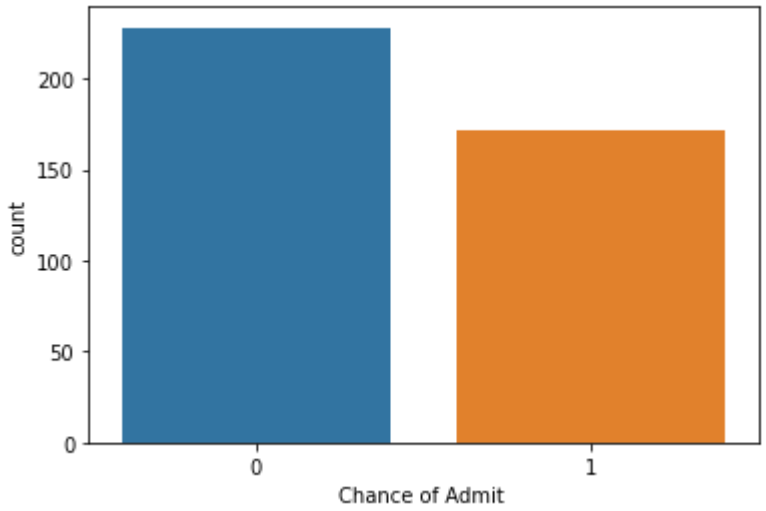
```
y
```

Out[20]:

```
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 [22]:

```
sns.countplot(x = y);
```



In [23]:

```
y.value_counts()
```

Out[23]:

```
0    228
1    172
Name: Chance of Admit , dtype: int64
```

In [26]:

```
#Crosss 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_size=0.25)
```

In [27]:

```
x_train.shape
```

Out[27]:

```
(300, 8)
```

In [28]:

```
x_test.shape
```

Out[28]:

```
(100, 8)
```

In [29]:

```
x_test
```

Out[29]:

	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 [30]:

```
x_train
```

Out[30]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research
250	251	320	104	3	3.0	2.5	8.57	1
63	64	315	107	2	4.0	3.0	8.50	1
312	313	311	107	4	4.5	4.5	9.00	1
159	160	297	100	1	1.5	2.0	7.90	0
283	284	321	111	3	2.5	3.0	8.90	1
...	...	...	...	...	...	...	...	...
323	324	305	102	2	2.0	2.5	8.18	0
192	193	322	114	5	4.5	4.0	8.94	1
117	118	290	104	4	2.0	2.5	7.46	0
47	48	339	119	5	4.5	4.0	9.70	0
172	173	322	110	4	4.0	5.0	9.13	1

300 rows × 8 columns

In [31]:

```
#import the Class
from sklearn.tree import DecisionTreeClassifier
```

In [32]:

```
classifier = DecisionTreeClassifier(random_state=0)
```

In [33]:

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

Out[33]:

```
DecisionTreeClassifier(random_state=0)
```

In [34]:

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

In [35]:

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

In [36]:

```
result
```

Out[36]:

	actual	predicted
132	0	0
309	0	0
341	1	1
196	0	0
246	0	1
...	...	...
146	0	0
135	1	1
390	0	0
264	0	0
364	1	1

100 rows × 2 columns

In [37]:

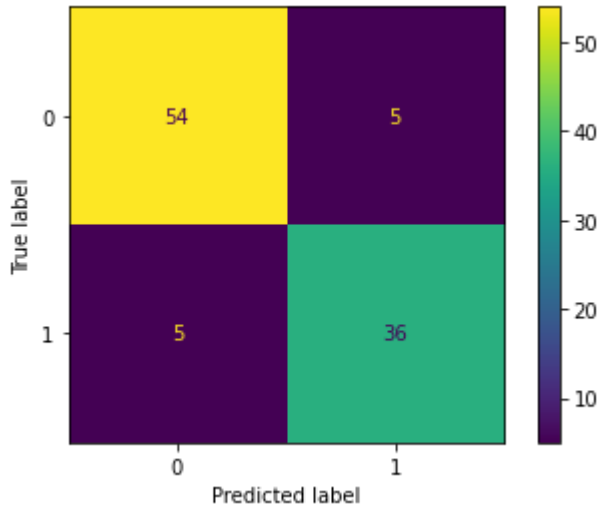
```
from sklearn.metrics import ConfusionMatrixDisplay, accuracy_score
from sklearn.metrics import classification_report
```

In [38]:

```
ConfusionMatrixDisplay.from_predictions(y_test,y_pred)
```

Out[38]:

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7f
9315bbc6a0>
```



In [39]:

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

Out[39]:

```
0.9
```

In [40]:

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

```
              precision    recall  f1-score   support

           0       0.92      0.92      0.92        59
           1       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 [41]:

```
new = [[284, 321, 111, 3, 2.5, 3.0, 8.90, 1]]
```

In [42]:

```
classifier.predict(new)[0]
```

```
/Users/shivraj/opt/anaconda3/lib/python3.9/site-packages/sklearn/base.
py:450: UserWarning: X does not have valid feature names, but Decision
TreeClassifier was fitted with feature names
  warnings.warn(
```

Out[42]:

```
1
```

In [45]:

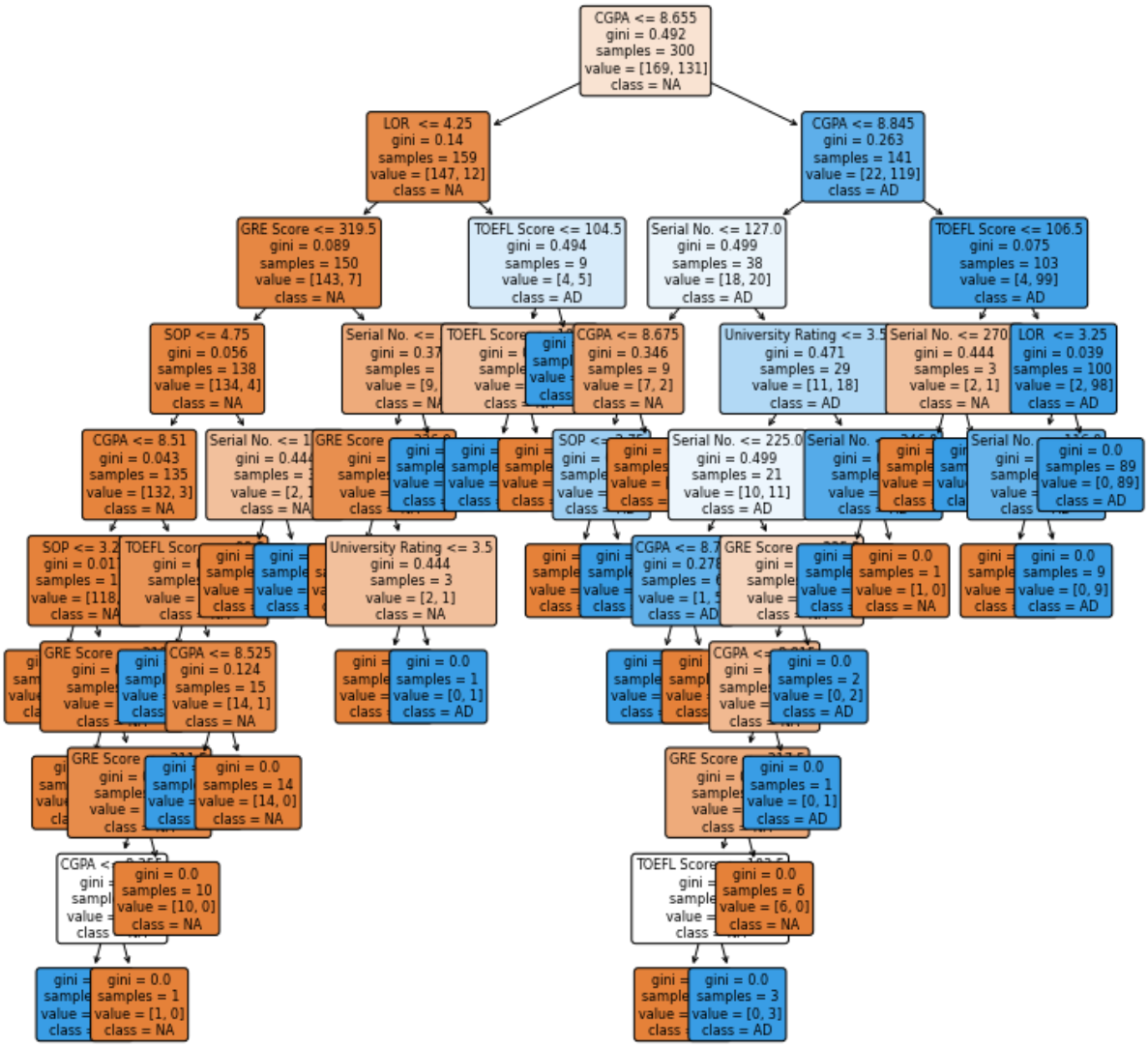
```
from sklearn.tree import plot_tree
plot_tree(classifier, )
```

Out[45]:

```
[Text(0.565625, 0.95, 'X[6] <= 8.655\ngini = 0.492\nsamples = 300\nval
ue = [169, 131]'),
 Text(0.365625, 0.85, 'X[5] <= 4.25\ngini = 0.14\nsamples = 159\nvalue
= [147, 12]'),
 Text(0.25625, 0.75, 'X[1] <= 319.5\ngini = 0.089\nsamples = 150\nvalu
e = [143, 7]'),
 Text(0.1625, 0.65, 'X[4] <= 4.75\ngini = 0.056\nsamples = 138\nvalue
= [134, 4]'),
 Text(0.1, 0.55, 'X[6] <= 8.51\ngini = 0.043\nsamples = 135\nvalue =
[132, 3]'),
 Text(0.05, 0.45, 'X[4] <= 3.25\ngini = 0.017\nsamples = 119\nvalue =
[118, 1]'),
 Text(0.025, 0.35, 'gini = 0.0\nsamples = 89\nvalue = [89, 0]'),
 Text(0.075, 0.35, 'X[1] <= 310.5\ngini = 0.064\nsamples = 30\nvalue =
[29, 1]'),
 Text(0.05, 0.25, 'gini = 0.0\nsamples = 18\nvalue = [18, 0]'),
 Text(0.1, 0.25, 'X[1] <= 311.5\ngini = 0.153\nsamples = 12\nvalue =
[11, 1]'),
```

In [49]:

```
import matplotlib.pyplot as plt
plt.figure(figsize=(12,12))
plot_tree(classifier,fontsize=8,filled=True,rounded=True,
          feature_names=x.columns,class_names=['NA','AD']);
```



In [ ]: