```
import pandas as pd
import seaborn as sns


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.columns
```

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

```
# Label encoding is not required here as the data in the frame is numeric d
from sklearn.preprocessing import Binarizer
bi = Binarizer(threshold=0.75)
df["Chance of Admit "] = bi.fit_transform(df[["Chance of Admit "]])
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	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
...	...	...	...	...	...	...	...	...	...
395	396	324	110	3	3.5	3.5	9.04	1	1.0
396	397	325	107	3	3.0	3.5	9.11	1	1.0
397	398	330	116	4	5.0	4.5	9.45	1	1.0
398	399	312	103	3	3.5	4.0	8.78	0	0.0
399	400	333	117	4	5.0	4.0	9.66	1	1.0

400 rows × 9 columns

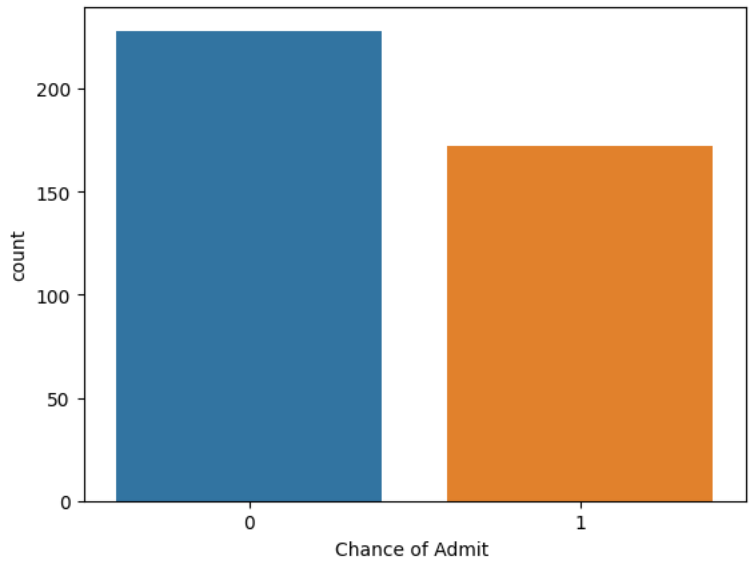
```
# set up of input and output data
x = df.drop("Chance of Admit ", axis=1)
y = df["Chance of Admit "]


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

```
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
```

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

```
<Axes: xlabel='Chance of Admit ', ylabel='count'>
```




```
y.value_counts()
```

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

```
# Cross Validation
from sklearn.model_selection import train_test_split

train_x, test_x, train_y, test_y = train_test_split(x,y, random_state=23)
```

```
train_x.shape
```

```
(300, 8)
```

```
test_x.shape
```

```
(100, 8)
```

```
from sklearn.tree import DecisionTreeClassifier
tree = DecisionTreeClassifier()
tree.fit(train_x, train_y)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

```
predicted = tree.predict(test_x)
```

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

```
result
```

	actual	predicted
133	1	1
331	0	0
167	0	0
335	1	1
239	0	0
...	...	...
280	0	1
392	1	1
44	1	1
221	0	0
363	0	0

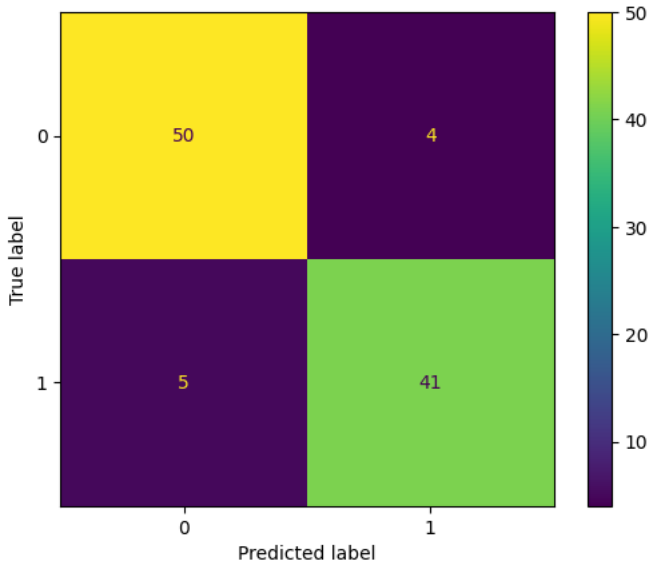
100 rows × 2 columns

```
from sklearn.metrics import classification_report

print(classification_report(result.actual, result.predicted))
```

```
                  precision    recall  f1-score   support

               0       0.91      0.93      0.92        54
               1       0.91      0.89      0.90        46

        accuracy                           0.91       100
       macro avg       0.91      0.91      0.91       100
    weighted avg       0.91      0.91      0.91       100
```

```
from sklearn.metrics import ConfusionMatrixDisplay, accuracy_score
ConfusionMatrixDisplay.from_predictions(result.actual, result.predicted)
```

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7b1b7afd9e70>
```




```
accuracy_score(result.actual, result.predicted)
```

```
0.91
```

```
import matplotlib.pyplot as plt
plt.figure(figsize=(20,20))
```

```
<Figure size 2000x2000 with 0 Axes>
<Figure size 2000x2000 with 0 Axes>
```