```
import pandas as pd


df=pd.read_csv("Admission_Predict.csv")


from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns


df.head()
```

	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

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
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

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
print(df.isnull().sum())
```

```
Serial No.           0
GRE Score            0
TOEFL Score          0
University Rating    0
SOP                  0
LOR                  0
CGPA                 0
Research             0
Chance of Admit      0
dtype: int64
```

```
df.columns
```

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

```
df.dtypes
```

	0
Serial No.	int64
GRE Score	int64
TOEFL Score	int64
University Rating	int64
SOP	float64
LOR	float64
CGPA	float64
Research	int64
Chance of Admit	float64

dtype: object

```
df = df.drop(columns=['Serial No.'])
```

```
print(df.isnull().sum())
```

```
GRE Score            0
TOEFL Score          0
University Rating    0
SOP                  0
LOR                  0
CGPA                 0
Research             0
Chance of Admit      0
dtype: int64
```

```
df['Chance of Admit '] = df['Chance of Admit '].apply(lambda x: 1 if x >= 0.75 else 0)
```

```
features = ['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'LOR ', 'CGPA', 'Research']
X = df[features]
y = df['Chance of Admit ']
```

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.3, random_state=42)
```

```
clf = DecisionTreeClassifier(random_state=42)
clf.fit(X_train, y_train)
```

```
DecisionTreeClassifier
DecisionTreeClassifier(random_state=42)
```

```
y_pred = clf.predict(X_test)
```

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

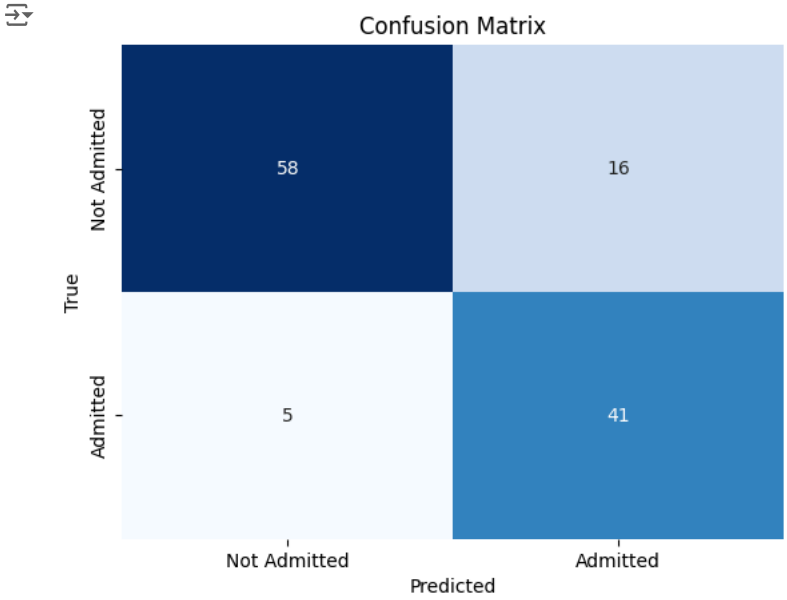
```
              precision    recall  f1-score   support

           0       0.92      0.78      0.85        74
           1       0.72      0.89      0.80        46

    accuracy                           0.82       120
   macro avg       0.82      0.84      0.82       120
weighted avg       0.84      0.82      0.83       120
```

```
cm = confusion_matrix(y_test, y_pred)
```

```
plt.figure(figsize=(8, 6))
```

```
<Figure size 800x600 with 0 Axes>
<Figure size 800x600 with 0 Axes>
```

```
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False,xticklabels=['Not Admitted', 'Admitted'],yticklabels=['Not Admitted', 'Ad
plt.xlabel('Predicted')
plt.ylabel('True')
plt.title('Confusion Matrix')
plt.show()
```



Start coding or generate with AI.