In [1]:
```
import pandas as pd
```

In [2]:
```
import numpy as np
```

In [3]:
```
df=pd.read_csv(r"C:\Users\DELL\Desktop\Admission_Predict.csv")
```

In [4]:
```
df
```

Out[4]:

	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 [5]:
```
df.head()
```

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

In [6]:
```
from sklearn.preprocessing import LabelEncoder
```

In [7]:
```
inputs=df.drop('Chance of Admit ',axis='columns')
```

In [8]:
```
target=df['Chance of Admit '].apply(lambda x:1 if x>0.6 else 0)
```

In [9]:
```
target
```

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

In [10]:
```
from sklearn.preprocessing import LabelEncoder
```

In [11]:
```
le_GRE=LabelEncoder()
```

In [12]:
```
le_CGPA=LabelEncoder()
```

In [13]:
```
inputs['GRE Score']=le_GRE.fit_transform(inputs['GRE Score'])
```

In [14]:
```
inputs['CGPA']=le_CGPA.fit_transform(inputs['GRE Score'])
```

In [15]:
```
inputs.head()
```

Out[15]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research
0	1	45	118	4	4.5	4.5	45	1
1	2	32	107	4	4.0	4.5	32	1
2	3	24	104	3	3.0	3.5	24	1
3	4	30	110	3	3.5	2.5	30	1
4	5	22	103	2	2.0	3.0	22	0

In [16]:
```
inputs =inputs.drop(['University Rating'],axis='columns')
inputs =inputs.drop(['TOEFL Score'],axis='columns')
inputs =inputs.drop(['SOP'],axis='columns')
inputs =inputs.drop(['LOR '],axis='columns')
inputs =inputs.drop(['Research'],axis='columns')
inputs =inputs.drop(['Serial No.'],axis='columns')
inputs
```

Out[16]:

	GRE Score	CGPA
0	45	45
1	32	32
2	24	24
3	30	30
4	22	22
...	...	...
395	32	32
396	33	33
397	38	38
398	20	20
399	41	41

400 rows × 2 columns

In [17]:
```
from sklearn.model_selection import train_test_split
```

In [18]:
```
x_train,x_test,y_train,y_test=train_test_split(inputs,target,test_size=0.25,random_state=True)
```

In [19]:
```
x_train.head()
```

Out[19]:

	GRE Score	CGPA
82	28	28
367	19	19
179	15	15
27	6	6
89	24	24

In [20]:
```
y_test.head()
```

Out[20]:
```
398    1
125    1
328    1
339    1
172    1
Name: Chance of Admit , dtype: int64
```

In [21]:
```
y_train.head()
```

Out[21]:
```
82     1
367    0
179    1
27     0
89     1
Name: Chance of Admit , dtype: int64
```

In [22]:
```
from sklearn import tree
```

In [23]:
```
model=tree.DecisionTreeClassifier()
```

In [24]:
```
model.fit(inputs,target)
```

Out[24]:
```
DecisionTreeClassifier()
```

In [25]:
```
model.predict([[316,8]])
```

```
C:\Users\DELL\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but DecisionTre
eClassifier was fitted with feature names
  warnings.warn(
```

Out[25]:
```
array([1], dtype=int64)
```

In [26]:
```
model.predict([[322,8.8]])
```

```
C:\Users\DELL\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X does not have valid feature names, but DecisionTre
eClassifier was fitted with feature names
  warnings.warn(
```

Out[26]:
```
array([1], dtype=int64)
```

In [27]:
```
pred=model.predict(x_test)
```

In [28]:
```
from sklearn.metrics import confusion_matrix
```

In [29]:
```
confusion_matrix(y_test,pred)
```

Out[29]:
```
array([[12, 10],
       [ 3, 75]], dtype=int64)
```

In [30]:
```
from sklearn.metrics import precision_score
precision_score(y_test,pred)
```

Out[30]:
```
0.8823529411764706
```

In [31]:
```
from sklearn.metrics import accuracy_score
accuracy_score(y_test,pred)
```

Out[31]:
```
0.87
```

In [32]:
```
from sklearn.metrics import recall_score
recall_score(y_test,pred)
```

Out[32]:
```
0.9615384615384616
```

In [33]:
```
from sklearn.metrics import f1_score
f1_score(y_test,pred)
```

Out[33]:
```
0.920245398773006
```

In [34]:
```
from sklearn.metrics import classification_report
print("Classification Report:\n",classification_report(y_test,pred))
```

```
Classification Report:
               precision    recall  f1-score   support

           0       0.80      0.55      0.65        22
           1       0.88      0.96      0.92        78

    accuracy                           0.87       100
   macro avg       0.84      0.75      0.78       100
weighted avg       0.86      0.87      0.86       100
```

In [35]:
```
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
print("MAR=",mean_squared_error(y_test,pred))
print("MAE=",mean_absolute_error(y_test,pred))
print("RMSE=",np.sqrt(mean_squared_error(y_test,pred)))
print("R2 Score=",r2_score(y_test,pred))
```

```
MAR= 0.13
MAE= 0.13
RMSE= 0.36055512754639896
R2 Score= 0.24242424242424254
```