In [1]:

```
import pandas as pd
import numpy as np
```

In [2]:

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

Out[2]:

	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 [3]:

```
df.shape
```

Out[3]:

```
(400, 9)
```

In [4]:

```
df.isnull()
```

Out[4]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...
395	False	False	False	False	False	False	False	False	False
396	False	False	False	False	False	False	False	False	False
397	False	False	False	False	False	False	False	False	False
398	False	False	False	False	False	False	False	False	False
399	False	False	False	False	False	False	False	False	False

400 rows × 9 columns

In [5]:

```
df.dtypes
```

Out[5]:

```
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
```

In [6]:

```
(df == 0).sum().sum()
```

Out[6]:

```
181
```

In [7]:

```
inputs = df.drop('Chance of Admit ',axis = 'columns')
target = df['Chance of Admit '].apply(lambda x : 1 if x > 0.6 else 0)
```

In [8]:

```
target
```

Out[8]:

```
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 [9]:

```
from sklearn.preprocessing import LabelEncoder
```

In [11]:

```
le_GRE = LabelEncoder()
le_CGPA = LabelEncoder()
```

In [12]:

```
inputs['GRE Score']=le_GRE.fit_transform(inputs['GRE Score'])
inputs['CGPA']=le_CGPA.fit_transform(inputs['CGPA'])
inputs.head()
```

Out[12]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research
0	1	45	118	4	4.5	4.5	156	1
1	2	32	107	4	4.0	4.5	101	1
2	3	24	104	3	3.0	3.5	32	1
3	4	30	110	3	3.5	2.5	84	1
4	5	22	103	2	2.0	3.0	49	0

In [13]:

```
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[13]:

	GRE Score	CGPA
0	45	156
1	32	101
2	24	32
3	30	84
4	22	49
...	...	...
395	32	113
396	33	119
397	38	145
398	20	95
399	41	157

400 rows × 2 columns

In [15]:

```
from sklearn.model_selection import train_test_split
X_train , X_test , Y_train , Y_test = train_test_split(inputs , target , test_size= 0.25, r
```

In [16]:

```
X_train.head()
```

Out[16]:

	GRE Score	CGPA
247	19	70
110	13	70
16	25	87
66	35	112
153	32	92

In [17]:

```
Y_test.head()
```

Out[17]:

```
209    1
280    1
33     1
210    1
93     0
Name: Chance of Admit , dtype: int64
```

In [18]:

```
Y_train.head()
```

Out[18]:

```
247    1
110    1
16     1
66     1
153    1
Name: Chance of Admit , dtype: int64
```

In [19]:

```
X_test.head()
```

Out[19]:

	GRE Score	CGPA
209	9	41
280	19	81
33	48	153
210	33	115
93	9	24

In [20]:

```
from sklearn import tree
```

In [21]:

```
model = tree.DecisionTreeClassifier()
```

In [22]:

```
model.fit(inputs, target)
```

Out[22]:

```
DecisionTreeClassifier()
```

In [23]:

```
model.predict([[316,8]])
```

```
C:\Users\HP\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but DecisionTreeClassifier was fitted wit
h feature names
  warnings.warn(
```

Out[23]:

```
array([1], dtype=int64)
```

In [24]:

```
model.predict([[322,8.8]])
```

```
C:\Users\HP\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarning: X
does not have valid feature names, but DecisionTreeClassifier was fitted wit
h feature names
  warnings.warn(
```

Out[24]:

```
array([1], dtype=int64)
```

In [25]:

```
pred = model.predict(X_test)
```

In [26]:

```
from sklearn.metrics import confusion_matrix
```

In [27]:

```
confusion_matrix(Y_test , pred)
```

Out[27]:

```
array([[21,  0],
       [ 1, 78]], dtype=int64)
```

In [30]:

```
from sklearn.metrics import accuracy_score
accuracy_score(Y_test , pred)
```

Out[30]:

```
0.99
```

In [31]:

```
from sklearn.metrics import precision_score
precision_score(Y_test , pred)
```

Out[31]:

```
1.0
```

In [32]:

```
from sklearn.metrics import recall_score
recall_score(Y_test , pred)
```

Out[32]:

```
0.9873417721518988
```

In [33]:

```
from sklearn.metrics import f1_score
f1_score(Y_test , pred)
```

Out[33]:

```
0.9936305732484078
```

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       0.95      1.00      0.98        21
           1       1.00      0.99      0.99        79

    accuracy                           0.99       100
   macro avg       0.98      0.99      0.99       100
weighted avg       0.99      0.99      0.99       100
```

In [35]:

```
from sklearn.metrics import mean_absolute_error
print("MAE=",mean_absolute_error(Y_test , pred))
```

```
MAE= 0.01
```

In [36]:

```
from sklearn.metrics import mean_squared_error
print ("MSE=", mean_squared_error(Y_test , pred))
```

```
MSE= 0.01
```

In [37]:

```
print ("RMSE=", np.sqrt(mean_squared_error(Y_test,pred)))
```

```
RMSE= 0.1
```

In [38]:

```
from sklearn.metrics import r2_score
r2 = r2_score(Y_test , pred)
print("R2_Score=", r2)
```

```
R2_Score= 0.9397227245328511
```

In [ ]:

```

```