In [3]:

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

In [4]:

```
dataset = pd.read_csv('student_scores.csv')
```

In [5]:

```
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, 1].values
```

In [6]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

In [7]:

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

Out[7]:

```
LinearRegression()
```

In [8]:

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

In [9]:

```
from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
Mean Absolute Error: 4.183859899002975
Mean Squared Error: 21.5987693072174
Root Mean Squared Error: 4.6474476121003665
```

In [1]:

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

In [3]:

```
dataset = pd.read_csv("User_Data.csv")
```

In [4]:

```
x = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values
```

In [7]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.20, random_state =
```

In [10]:

```
from sklearn.preprocessing import StandardScaler
sc_x = StandardScaler()
xtrain = sc_x.fit_transform(X_train)
xtest = sc_x.transform(X_test)
```

In [12]:

```
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state = 0)
classifier.fit(xtrain, y_train)
```

Out[12]:

```
LogisticRegression(random_state=0)
```

In [13]:

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

In [14]:

```
y_pred
```

Out[14]:

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

In [16]:

```
from sklearn.metrics import accuracy_score
print ("Accuracy : ", accuracy_score(y_test, y_pred))
```

```
Accuracy :  1.0
```

In [1]:

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

In [2]:

```
dataset = pd.read_csv('house_data.csv')
```

In [15]:

```
dataset.shape
```

Out[15]:

```
(21613, 11)
```

In [4]:

```
X = dataset.iloc[:,[3,4]].values
```

In [5]:

```
y = dataset.iloc[:, -1].values
```

In [6]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

In [7]:

```
y
```

Out[7]:

```
array([221900., 538000., 180000., ..., 402101., 400000., 325000.])
```

In [8]:

```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

Out[8]:

```
LinearRegression()
```

In [9]:

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

In [10]:

```
y_pred
```

Out[10]:

```
array([795554.35059871, 660789.55506513, 446053.54464655, ...,
       621652.28245232, 534707.75526181, 452804.43542934])
```

In [11]:

```
from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
Mean Absolute Error: 217846.7647697289
Mean Squared Error: 109993546353.38124
Root Mean Squared Error: 331652.74965448614
```

In [ ]:

In [1]:

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

In [2]:

```
dataset = pd.read_csv("User_Data.csv")
```

In [3]:

```
x = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values
```

In [4]:

```
ı sklearn.model_selection import train_test_split
ain, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.20, random_state = 0)
```

In [5]:

```
from sklearn.preprocessing import StandardScaler
sc_x = StandardScaler()
xtrain = sc_x.fit_transform(X_train)
xtest = sc_x.transform(X_test)
```

In [6]:

```
from sklearn.neighbors import KNeighborsClassifier
classifier = KNeighborsClassifier(n_neighbors=5)
classifier.fit(xtrain, y_train)
```

Out[6]:

```
KNeighborsClassifier()
```

In [7]:

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

In [8]:

```
y_pred
```

Out[8]:

```
array([0, 0, 0, 0], dtype=int64)
```

In [9]:

```
#Creating the Confusion matrix
from sklearn.metrics import classification_report,confusion_matrix
cm= confusion_matrix(y_test, y_pred)
cm
```

Out[9]:

```
array([[3, 0],
       [1, 0]], dtype=int64)
```

In [10]:

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

```
              precision    recall  f1-score   support

           0       0.75      1.00      0.86         3
           1       0.00      0.00      0.00         1

    accuracy                           0.75         4
   macro avg       0.38      0.50      0.43         4
weighted avg       0.56      0.75      0.64         4


C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.p
y:1245: UndefinedMetricWarning: Precision and F-score are ill-defined and be
ing set to 0.0 in labels with no predicted samples. Use `zero_division` para
meter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.p
y:1245: UndefinedMetricWarning: Precision and F-score are ill-defined and be
ing set to 0.0 in labels with no predicted samples. Use `zero_division` para
meter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\ProgramData\Anaconda3\lib\site-packages\sklearn\metrics\_classification.p
y:1245: UndefinedMetricWarning: Precision and F-score are ill-defined and be
ing set to 0.0 in labels with no predicted samples. Use `zero_division` para
meter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [ ]:

In [1]:

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

In [2]:

```
dataset = pd.read_csv("User_Data.csv")
```

In [3]:

```
x = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values
```

In [4]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.20, random_state =
```

In [5]:

```
from sklearn.preprocessing import StandardScaler
sc_x = StandardScaler()
xtrain = sc_x.fit_transform(X_train)
xtest = sc_x.transform(X_test)
```

In [6]:

```
from sklearn.svm import SVC # "Support vector classifier"
classifier = SVC(kernel='linear', random_state=0)
classifier.fit(xtrain, y_train)
```

Out[6]:

```
SVC(kernel='linear', random_state=0)
```

In [7]:

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

In [8]:

```
y_pred
```

Out[8]:

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

In [9]:

```
#Creating the Confusion matrix
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(y_test, y_pred)
cm
```

Out[9]:

```
array([[3, 0],
       [0, 1]], dtype=int64)
```

In [10]:

```
from sklearn.metrics import accuracy_score
print ("Accuracy : ", accuracy_score(y_test, y_pred))
```

```
Accuracy :  1.0
```

In [1]:

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

In [2]:

```
dataset = pd.read_csv("User_Data.csv")
```

In [3]:

```
x = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values
```

In [4]:

```
ı sklearn.model_selection import train_test_split
rain, X_test, y_train, y_test = train_test_split(x, y, test_size = 0.20, random_state = 0)
```

In [5]:

```
from sklearn.preprocessing import StandardScaler
sc_x = StandardScaler()
xtrain = sc_x.fit_transform(X_train)
xtest = sc_x.transform(X_test)
```

In [6]:

```
#Fitting Decision Tree classifier to the training set
from sklearn.tree import DecisionTreeClassifier
classifier= DecisionTreeClassifier(criterion='entropy', random_state=0)
classifier.fit(xtrain, y_train)
```

Out[6]:

```
DecisionTreeClassifier(criterion='entropy', random_state=0)
```

In [7]:

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

In [8]:

```
y_pred
```

Out[8]:

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

In [9]:

```
#Creating the Confusion matrix
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(y_test, y_pred)
cm
```

Out[9]:

```
array([[3, 0],
       [0, 1]], dtype=int64)
```

In [10]:

```
from sklearn.metrics import accuracy_score
print ("Accuracy : ", accuracy_score(y_test, y_pred))
```

```
Accuracy :  1.0
```

In [ ]: