1.

```
import numpy as np
import matplotlib.pyplot as plt

#Y=X+2
X = np.array([2,3,4,5,6,7,8,9,10])
Y = np.array([4,5,6,7,8,9,10,11,12])

len(X)
```

```
9
```

```
X = X.reshape(-1,1)
Y = Y.reshape(-1,1)

X
```

```
array([[ 2],
       [ 3],
       [ 4],
       [ 5],
       [ 6],
       [ 7],
       [ 8],
       [ 9],
       [10]])
```

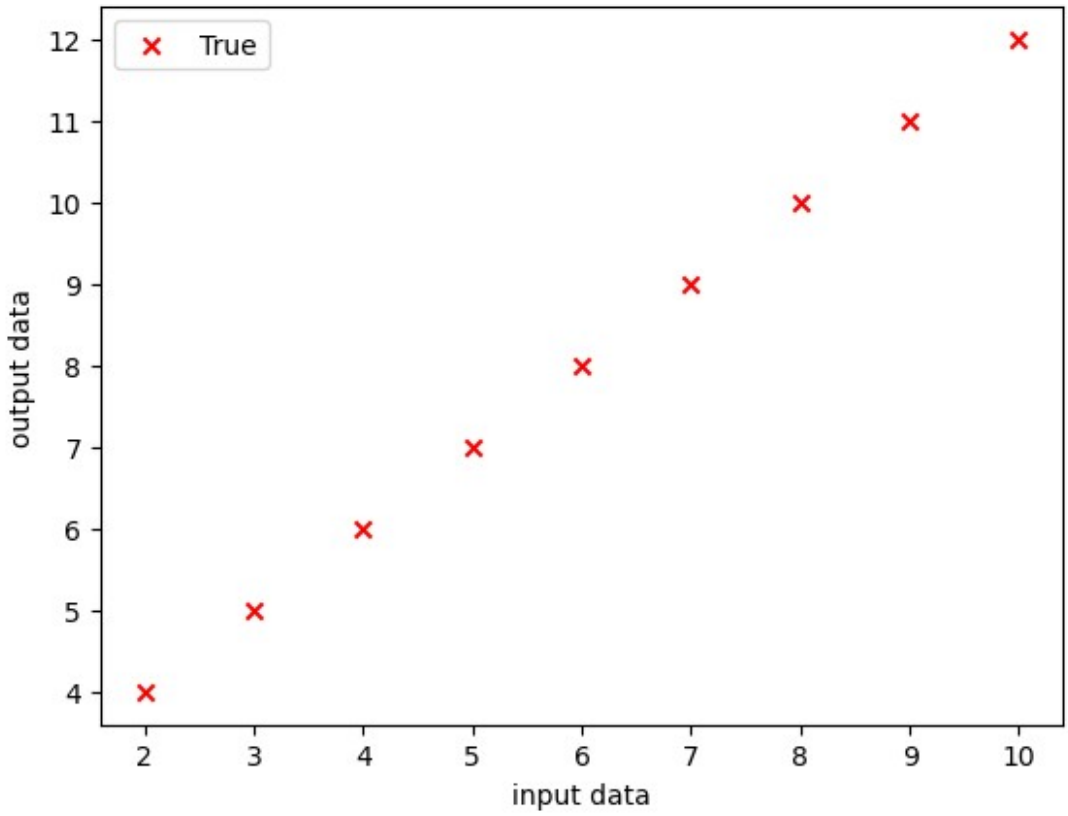
```
print(f"size of input data : {len(X)}")
```

```
size of input data : 9
```

```
#(2,4) (3,5)
for i in range(len(X)):
    print((X[i],Y[i]))
```

```
(2, 4)
(3, 5)
(4, 6)
(5, 7)
(6, 8)
(7, 9)
(8, 10)
(9, 11)
(10, 12)
```

```
plt.scatter(X,Y,marker='x',c='red',label='True')
plt.xlabel("input data")
plt.ylabel("output data")
plt.legend()
plt.show()
```




```
#y=x**2+1
x = np.array([2,3,4,5,6,7,8,9,10])
y= np.array([5,10,17,26,37,50,65,82,101])

print(f"size of x is: {len(x)}")
```

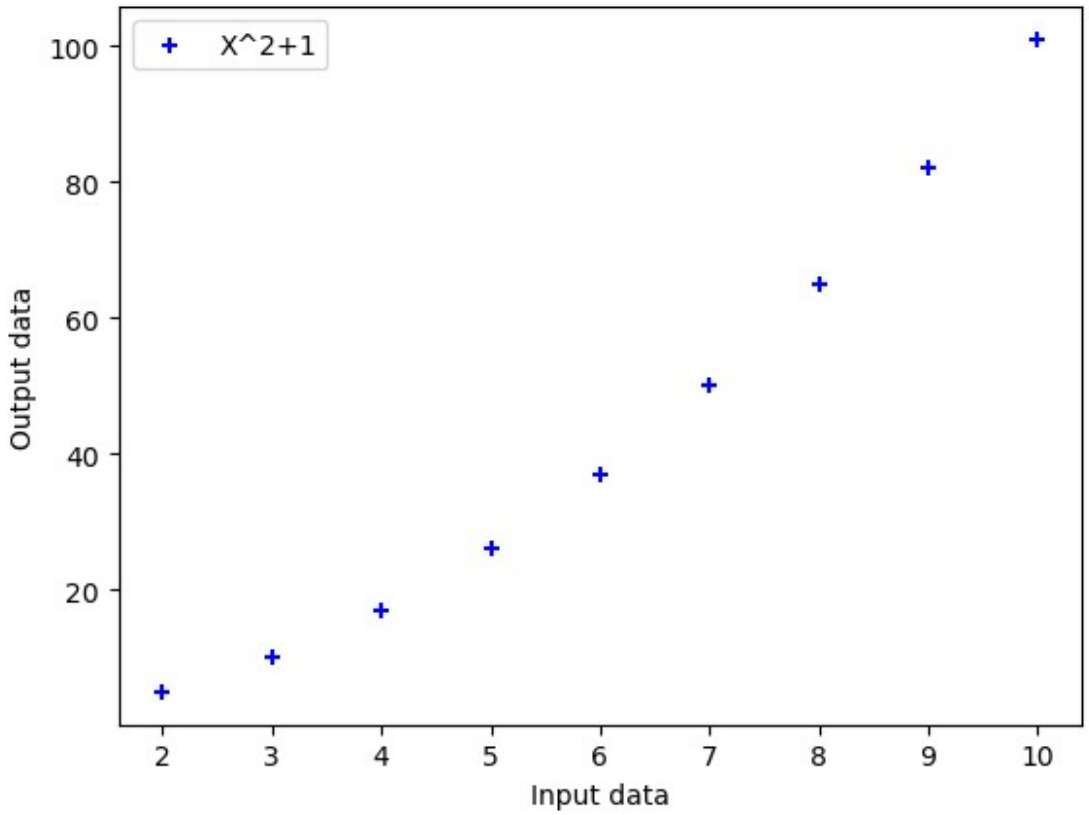
```
size of x is: 9
```

```
for i in range (len(x)):
    print((x[i],y[i]))
```

```
(2, 5)
(3, 10)
(4, 17)
(5, 26)
(6, 37)
(7, 50)
(8, 65)
(9, 82)
(10, 101)
```

```
plt.scatter(x,y,marker='+',c='b',label='X^2+1')
plt.xlabel("Input data")
plt.ylabel("Output data")
```

```
plt.legend()
plt.show()
```




```
pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in
/opt/anaconda3/lib/python3.12/site-packages (1.5.1)
Requirement already satisfied: numpy>=1.19.5 in
/opt/anaconda3/lib/python3.12/site-packages (from scikit-learn)
(1.26.4)
Requirement already satisfied: scipy>=1.6.0 in
/opt/anaconda3/lib/python3.12/site-packages (from scikit-learn)
(1.13.1)
Requirement already satisfied: joblib>=1.2.0 in
/opt/anaconda3/lib/python3.12/site-packages (from scikit-learn)
(1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in
/opt/anaconda3/lib/python3.12/site-packages (from scikit-learn)
(3.5.0)
Note: you may need to restart the kernel to use updated packages.
```

```
from sklearn.linear_model import LinearRegression

model_LR =LinearRegression()
```

```
model_LR.fit(X,Y)
```

```
LinearRegression()
```

```
model_LR.predict([[11]])
```

```
array([[13.]])
```

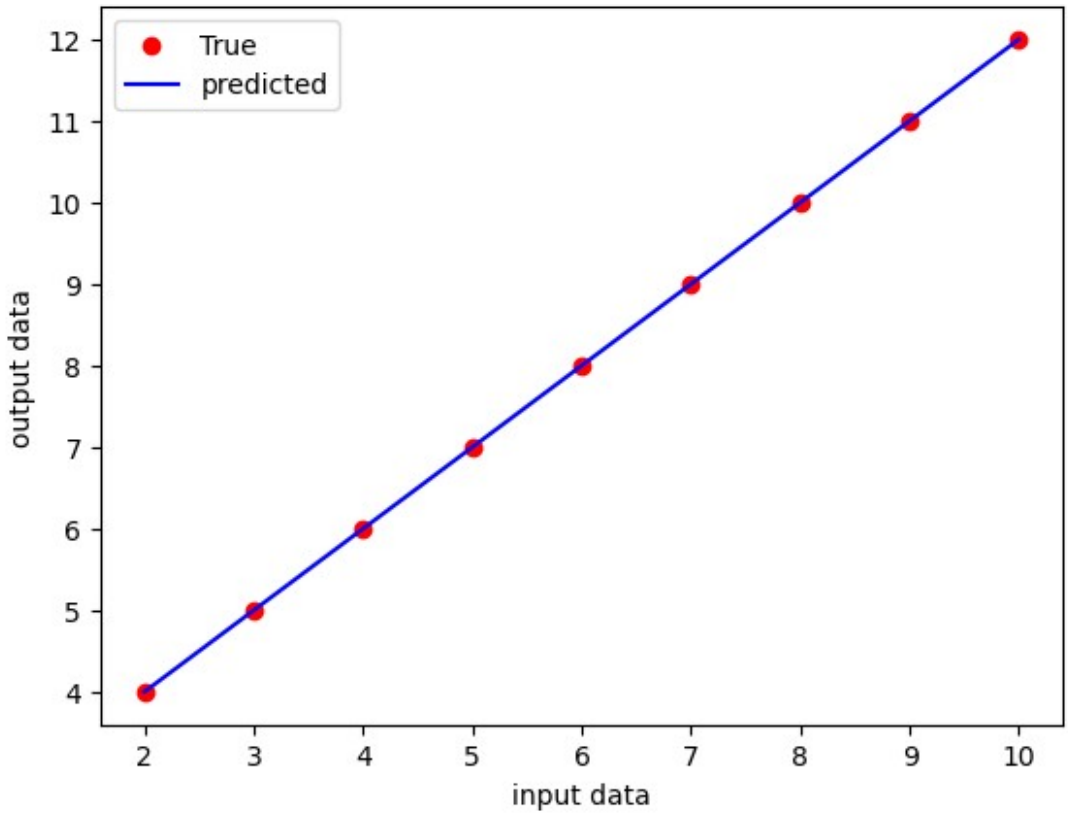
```
test_x = np.array([2,3,4,5,6,7,8,9,10]).reshape(-1,1)
```

```
predict_y = model_LR.predict(test_x)
```

```
predict_y
```

```
array([[ 4.],
       [ 5.],
       [ 6.],
       [ 7.],
       [ 8.],
       [ 9.],
       [10.],
       [11.],
       [12.]])
```

```
plt.scatter(X,Y,marker='o',c='red',label='True')
plt.plot(test_x,predict_y,c='blue',label='predicted')
plt.xlabel("input data")
plt.ylabel("output data")
plt.legend()
plt.show()
```




```
#y=x**2+1
X = np.array([2,3,4,5,6,7,8,9,10])
Y= np.array([5,10,17,26,37,50,65,82,101])

print(f"size of x is: {len(x)}")
```

```
size of x is: 9
```

```
X = X.reshape(-1,1)
Y = Y.reshape(-1,1)
X
```

```
array([[ 2],
       [ 3],
       [ 4],
       [ 5],
       [ 6],
       [ 7],
       [ 8],
       [ 9],
       [10]])
```

```
model_LR =LinearRegression()

model_LR.fit(X,Y)
```

```
LinearRegression()
```

```
model_LR.predict([[11]])
```

```
array([[103.66666667]])
```

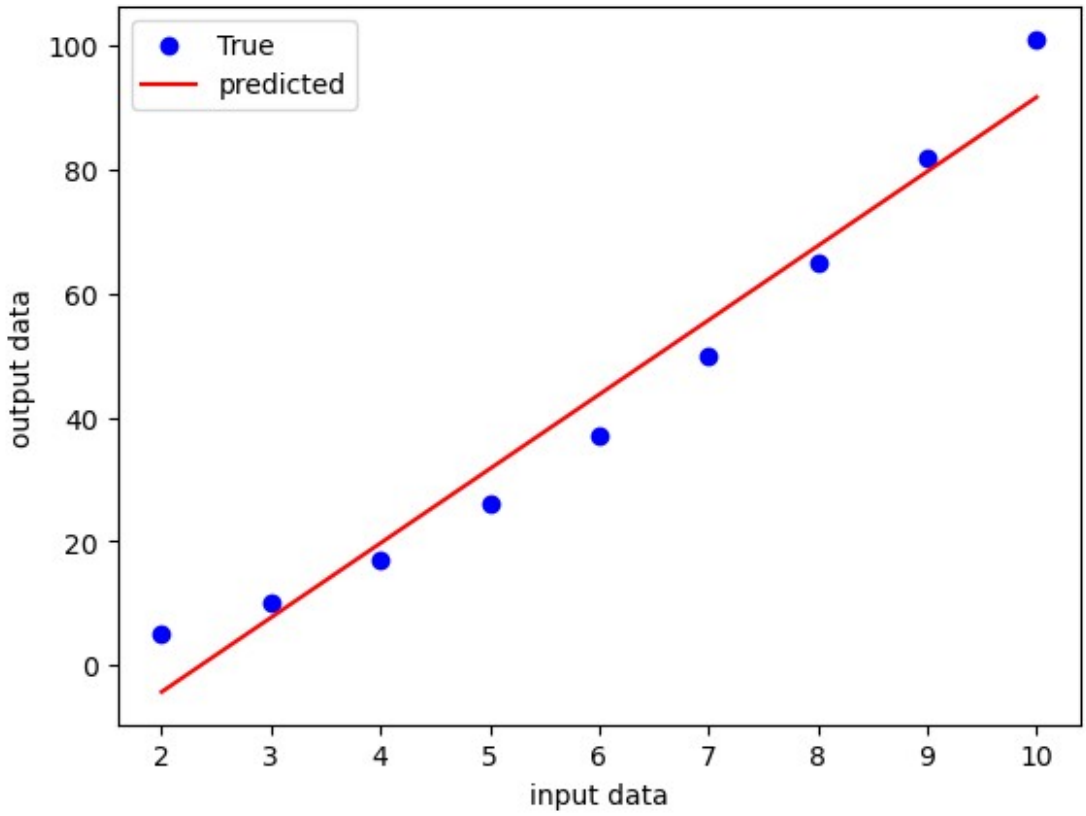
```
test_x = np.array([2,3,4,5,6,7,8,9,10]).reshape(-1,1)
```

```
predict_y = model_LR.predict(test_x)
```

```
predict_y
```

```
array([[-4.33333333],
       [ 7.66666667],
       [19.66666667],
       [31.66666667],
       [43.66666667],
       [55.66666667],
       [67.66666667],
       [79.66666667],
       [91.66666667]])
```

```
plt.scatter(X,Y,marker='o',c='blue',label='True')
plt.plot(test_x,predict_y,c='red',label='predicted')
plt.xlabel("input data")
plt.ylabel("output data")
plt.legend()
plt.show()
```


True
predicted
output data
input data
100
80
60
40
20
0
2
3
4
5
6
7
8
9
10

2.

```
import pandas as pd

import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

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

df
```

```
        price  area  bedrooms  bathrooms  stories mainroad guestroom
basement  \
0    13300000  7420         4          2        3      yes        no
no
1    12250000  8960         4          4        4      yes        no
no
2    12250000  9960         3          2        2      yes        no
yes
3    12215000  7500         4          2        2      yes        no
yes
4    11410000  7420         4          1        2      yes       yes
yes
..        ...   ...       ...        ...      ...      ...       ...
...
540   1820000  3000         2          1        1      yes        no
yes
541   1767150  2400         3          1        1       no        no
no
542   1750000  3620         2          1        1      yes        no
no
543   1750000  2910         3          1        1       no        no
no
544   1750000  3850         3          1        2      yes        no
no

    hotwaterheating airconditioning  parking prefarea furnishingstatus

0                no             yes        2      yes        furnished

1                no             yes        3       no        furnished

2                no              no        2      yes   semi-furnished

3                no             yes        3      yes        furnished

4                no             yes        2       no        furnished

..              ...             ...      ...      ...              ...

540              no              no        2       no      unfurnished
```

```
541               no               no         0        no    semi-furnished

542               no               no         0        no       unfurnished

543               no               no         0        no         furnished

544               no               no         0        no       unfurnished
```

```
X=df['area']
```

```
X
```

```
0      7420
1      8960
2      9960
3      7500
4      7420
       ...
540    3000
541    2400
542    3620
543    2910
544    3850
Name: area, Length: 545, dtype: int64
```

```
Y=df['price']
```

```
Y
```

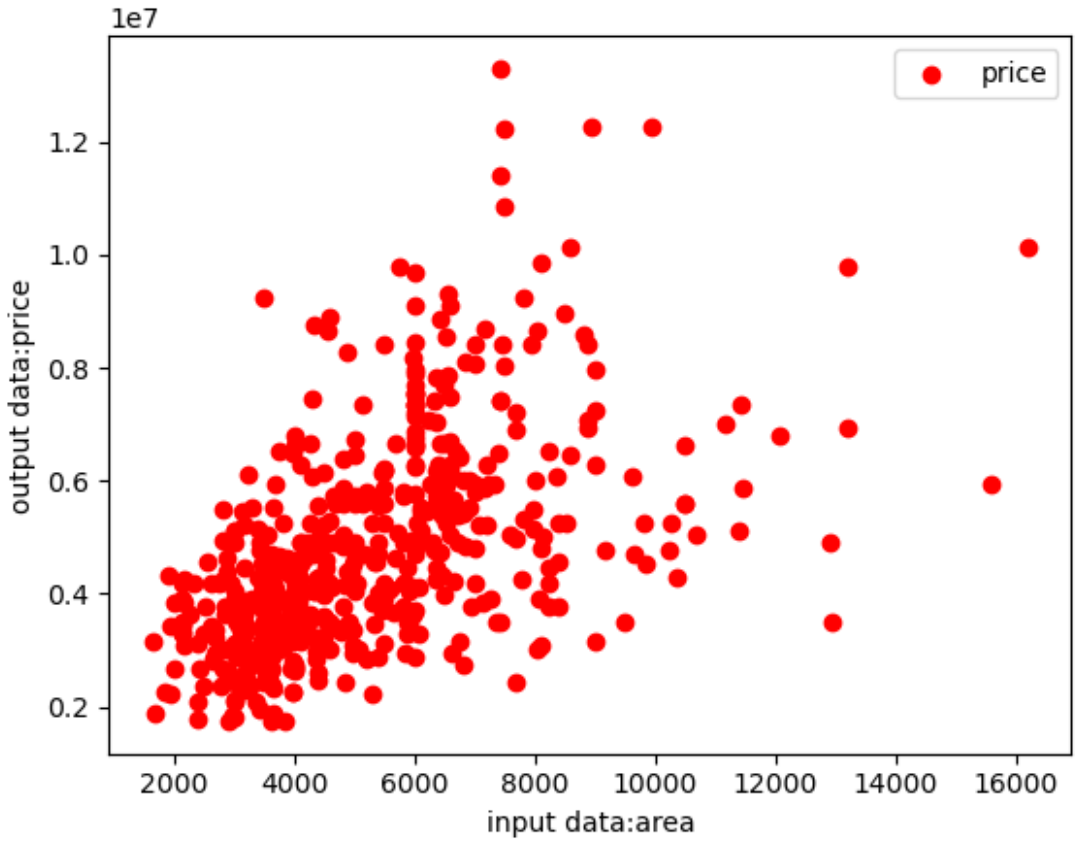
```
0      13300000
1      12250000
2      12250000
3      12215000
4      11410000
         ...
540     1820000
541     1767150
542     1750000
543     1750000
544     1750000
Name: price, Length: 545, dtype: int64
```

```
X=np.array(X).reshape(-1,1)
Y=np.array(Y).reshape(-1,1)
```

```
X
```

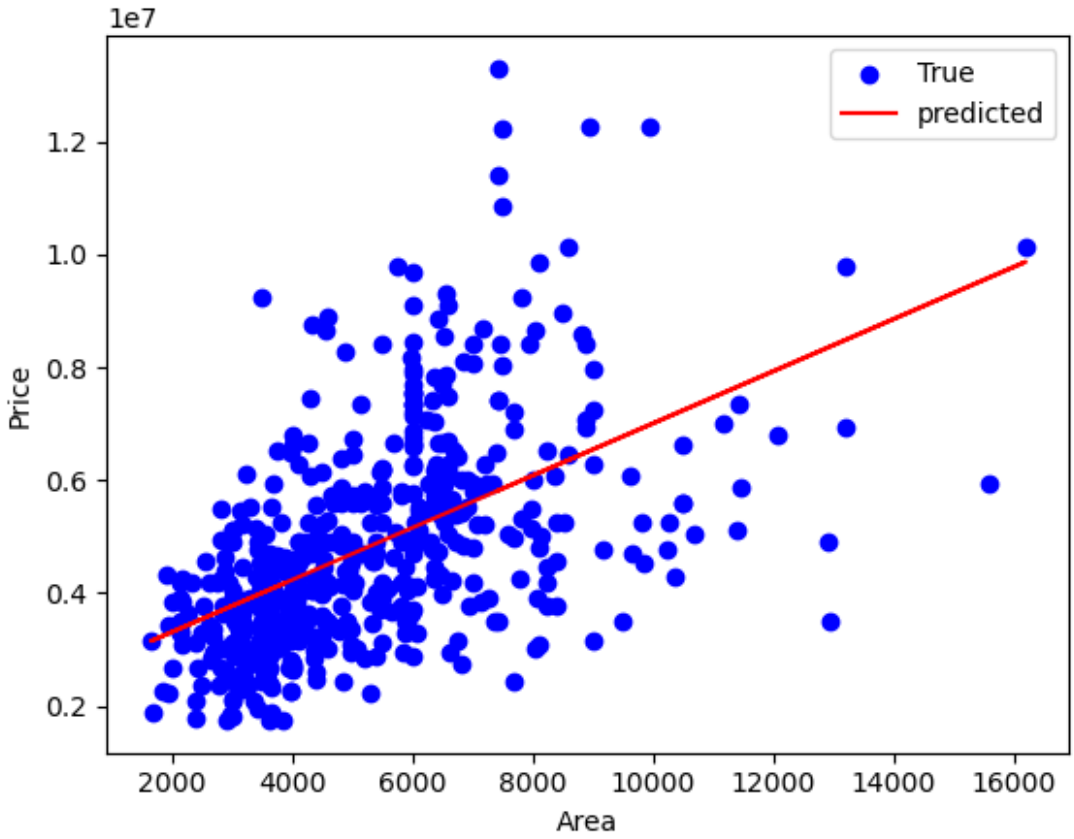
```
array([[ 7420],
       [ 8960],
       [ 9960],
```

```
       [ 2910],
       [ 3850]])

print(f"size of input data : {len(X)}")

size of input data : 545

plt.scatter(X,Y,marker='o',c='red',label='price')
plt.xlabel("input data:area")
plt.ylabel("output data:price")
plt.legend()
plt.show()
```




```
#LINEAR REGRESSION MODEL
model_LR = LinearRegression()

model_LR.fit(X,Y) #providing X & Y to the model

LinearRegression()

model_LR.predict([[2500]]) #predicting

array([[3542245.71807839]])
```

```
predict_y = model_LR.predict(X)

plt.scatter(X,Y,c="blue",label="True")
plt.plot(X,predict_y,c="red",label="predicted")
plt.xlabel("Area")
plt.ylabel("Price")
plt.legend()
plt.show()
```



3.

```
import pandas as pd

import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split

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

df
```

```
        price  area  bedrooms  bathrooms  stories mainroad guestroom
basement  \
0    13300000  7420         4          2        3      yes        no
no
1    12250000  8960         4          4        4      yes        no
no
2    12250000  9960         3          2        2      yes        no
yes
3    12215000  7500         4          2        2      yes        no
yes
4    11410000  7420         4          1        2      yes       yes
yes
..        ...   ...       ...        ...      ...      ...       ...
...
540   1820000  3000         2          1        1      yes        no
yes
541   1767150  2400         3          1        1       no        no
no
542   1750000  3620         2          1        1      yes        no
no
543   1750000  2910         3          1        1       no        no
no
544   1750000  3850         3          1        2      yes        no
no

    hotwaterheating airconditioning  parking prefarea furnishingstatus

0                no             yes        2      yes        furnished

1                no             yes        3       no        furnished

2                no              no        2      yes   semi-furnished

3                no             yes        3      yes        furnished

4                no             yes        2       no        furnished

..              ...             ...      ...      ...              ...

540              no              no        2       no      unfurnished
```

```
540              no                no        2        no      unfurnished

541              no                no        0        no   semi-furnished

542              no                no        0        no      unfurnished

543              no                no        0        no        furnished

544              no                no        0        no      unfurnished
```

```
X=df['area']
```

```
X
```

```
0      7420
1      8960
2      9960
3      7500
4      7420
       ...
540    3000
541    2400
542    3620
543    2910
544    3850
Name: area, Length: 545, dtype: int64
```

```
Y=df['price']
```

```
Y
```

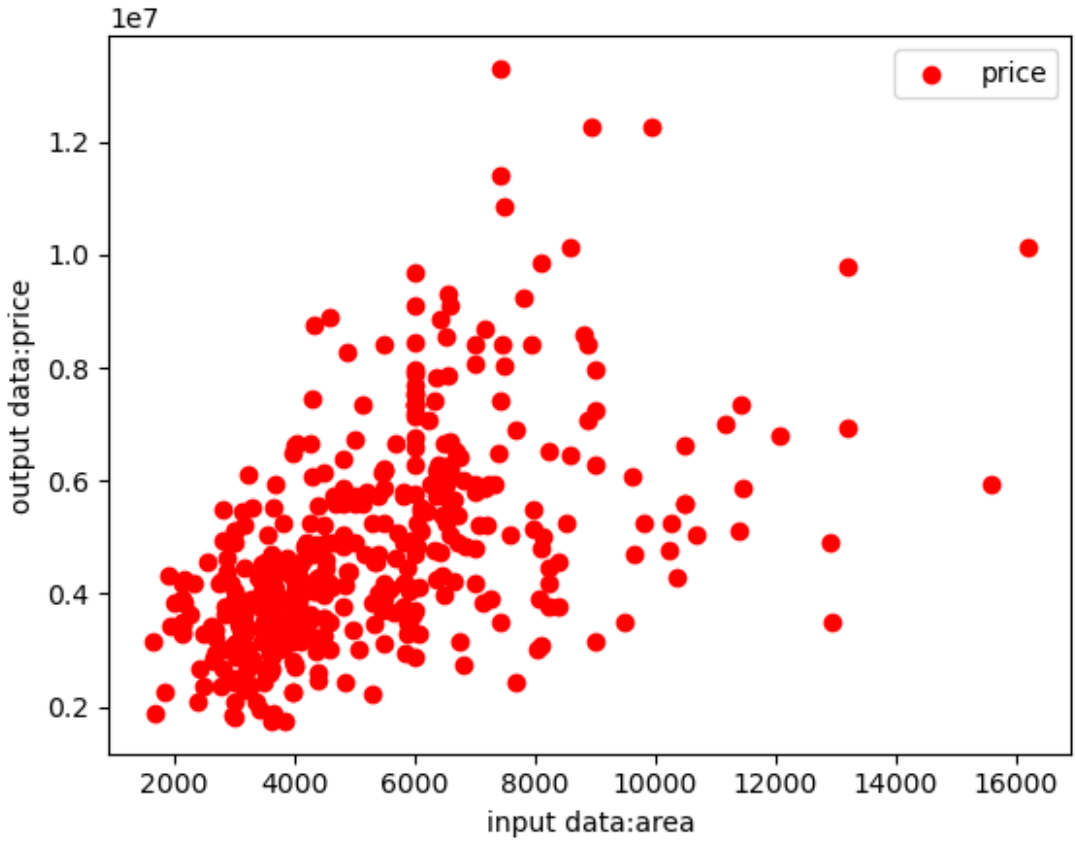
```
0      13300000
1      12250000
2      12250000
3      12215000
4      11410000
         ...
540     1820000
541     1767150
542     1750000
543     1750000
544     1750000
Name: price, Length: 545, dtype: int64
```

```
X=np.array(X).reshape(-1,1)
Y=np.array(Y).reshape(-1,1)

#using sklearn split dataset
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size = 0.2)
```

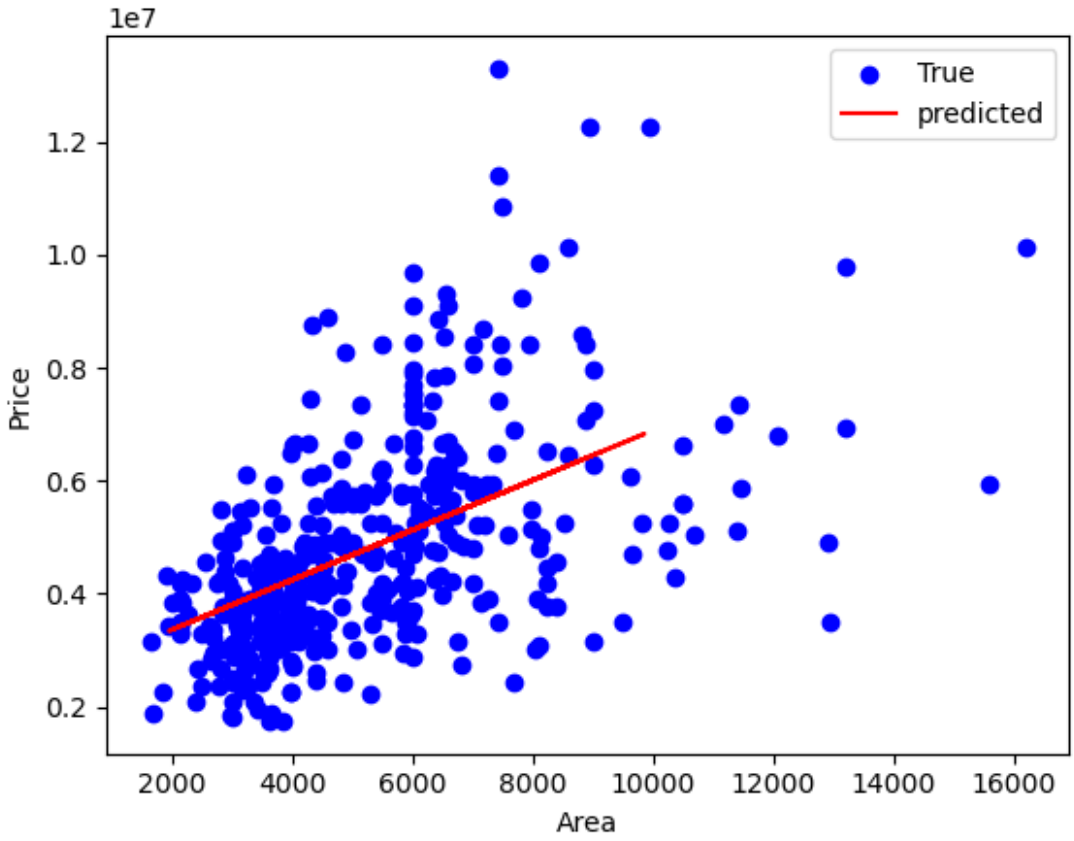
```
'''
x_train = X[0:435]
y_train = Y[0:435]
x_test = X[435:]
y_test = Y[435:]
'''
#manually
X_train = X[0:435]
Y_train = Y[0:435]
X_test = X[435:]
Y_test = Y[435:]

print(f"size of training data of X : {len(X_train)}")
print(f"size of testing data of X : {len(X_test)}")
print(f"size of training data of Y : {len(Y_train)}")
print(f"size of testing data of Y : {len(Y_test)}")
```

```
size of training data of X : 436
size of testing data of X : 109
size of training data of Y : 436
size of testing data of Y : 109
```

```
plt.scatter(X_train,Y_train,marker='o',c='red',label='price')
plt.xlabel("input data:area")
plt.ylabel("output data:price")
plt.legend()
plt.show()
```




```
#LINEAR REGRESSION MODEL
model_LR = LinearRegression()

model_LR.fit(X_train,Y_train) #providing X & Y to the model

LinearRegression()

model_LR.predict([[2500]]) #predicting

array([[3585124.1100557]])

predict_y = model_LR.predict(X_test)

plt.scatter(X_train,Y_train,c="blue",label="True")
plt.plot(X_test,predict_y,c="red",label="predicted")
plt.xlabel("Area")
plt.ylabel("Price")
plt.legend()
plt.show()
```


1e7
True
predicted
1.2
1.0
0.8
0.6
0.4
0.2
Price
2000
4000
6000
8000
10000
12000
14000
16000
Area

```
import pandas as pd

import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split

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

df
```

```
      Class  Tumor_Size Unnamed: 2
0         0   98.613971        NaN
1         0   63.858816        NaN
2         0   81.867206          s
3         1  151.229741        NaN
4         1  174.988756        NaN
...     ...         ...        ...
3743      1  158.437600        NaN
3744      1  161.158675        NaN
3745      1  167.130118        NaN
3746      1  223.812932        NaN
3747      1  239.251388        NaN

[3748 rows x 3 columns]
```

```
df.head()
```

```
   Class  Tumor_Size Unnamed: 2
0      0   98.613971        NaN
1      0   63.858816        NaN
2      0   81.867206          s
3      1  151.229741        NaN
4      1  174.988756        NaN
```

```
df.tail()
```

```
      Class  Tumor_Size Unnamed: 2
3743      1  158.437600        NaN
3744      1  161.158675        NaN
3745      1  167.130118        NaN
3746      1  223.812932        NaN
3747      1  239.251388        NaN
```

```
X=df['Tumor_Size']

X
```

```
0        98.613971
1        63.858816
2        81.867206
3       151.229741
```

```
4       174.988756
           ...
3743    158.437600
3744    161.158675
3745    167.130118
3746    223.812932
3747    239.251388
Name: Tumor_Size, Length: 3748, dtype: float64
```

```
Y=df['Class']
```

```
Y
```

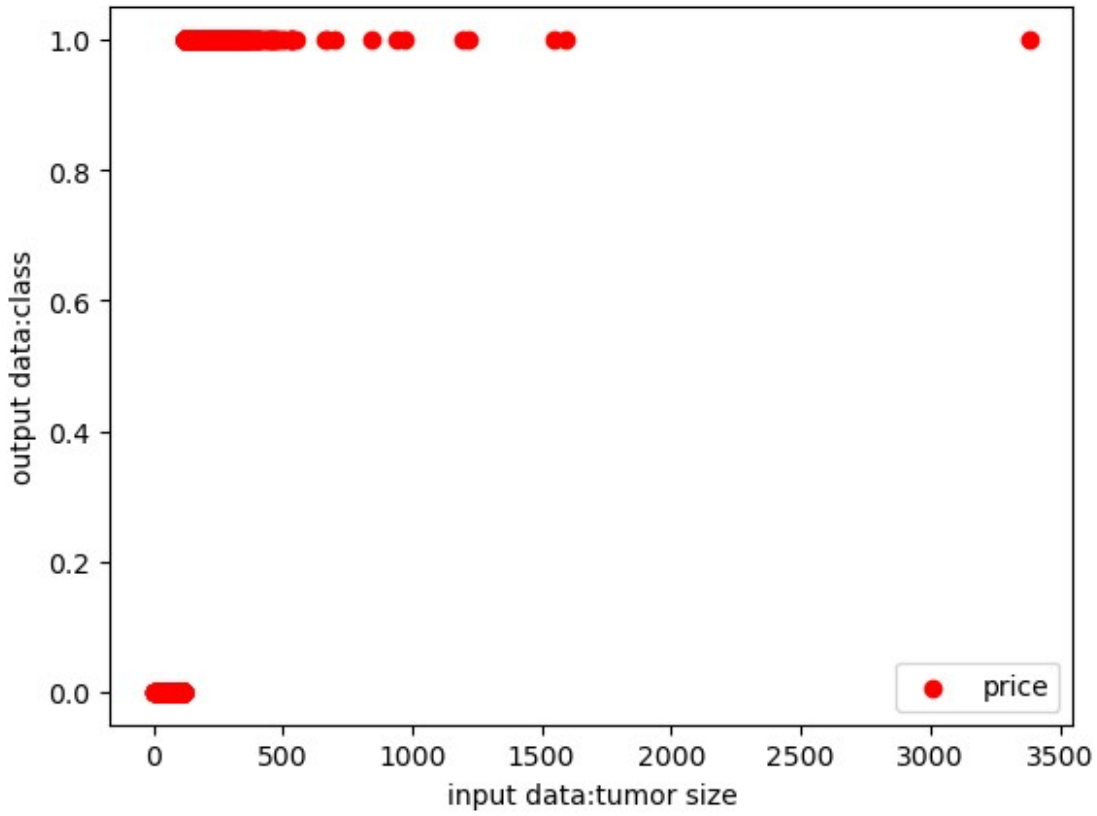
```
0       0
1       0
2       0
3       1
4       1
       ..
3743    1
3744    1
3745    1
3746    1
3747    1
Name: Class, Length: 3748, dtype: int64
```

```
X=np.array(X).reshape(-1,1)
Y=np.array(Y).reshape(-1,1)

#using sklearn split dataset
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size = 0.2)

'''
x_train = X[0:435]
y_train = Y[0:435]
x_test = X[435:]
y_test = Y[435:]

#manually
X_train = X[0:435]
Y_train = Y[0:435]
X_test = X[435:]
Y_test = Y[435:]
'''

'\nx_train = X[0:435]\ny_train = Y[0:435]\nx_test = X[435:]\ny_test =
Y[435:]\n\n#manually\nX_train = X[0:435]\nY_train = Y[0:435]\nX_test =
X[435:]\nY_test = Y[435:]\n'

print(f"size of training data of X : {len(X_train)}")
print(f"size of testing data of X : {len(X_test)}")
```

```
print(f"size of training data of Y : {len(Y_train)}")
print(f"size of testing data of Y : {len(Y_test)}")
```

```
size of training data of X : 2998
size of testing data of X : 750
size of training data of Y : 2998
size of testing data of Y : 750
```

```
plt.scatter(X_train,Y_train,marker='o',c='red',label='price')
plt.xlabel("input data:tumor size")
plt.ylabel("output data:class")
plt.legend()
plt.show()
```




```
#LINEAR REGRESSION MODEL
model_LR = LogisticRegression()

model_LR.fit(X_train,Y_train) #providing X & Y to the model
```

```
/opt/anaconda3/lib/python3.12/site-packages/sklearn/utils/
validation.py:1339: DataConversionWarning: A column-vector y was
passed when a 1d array was expected. Please change the shape of y to
(n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

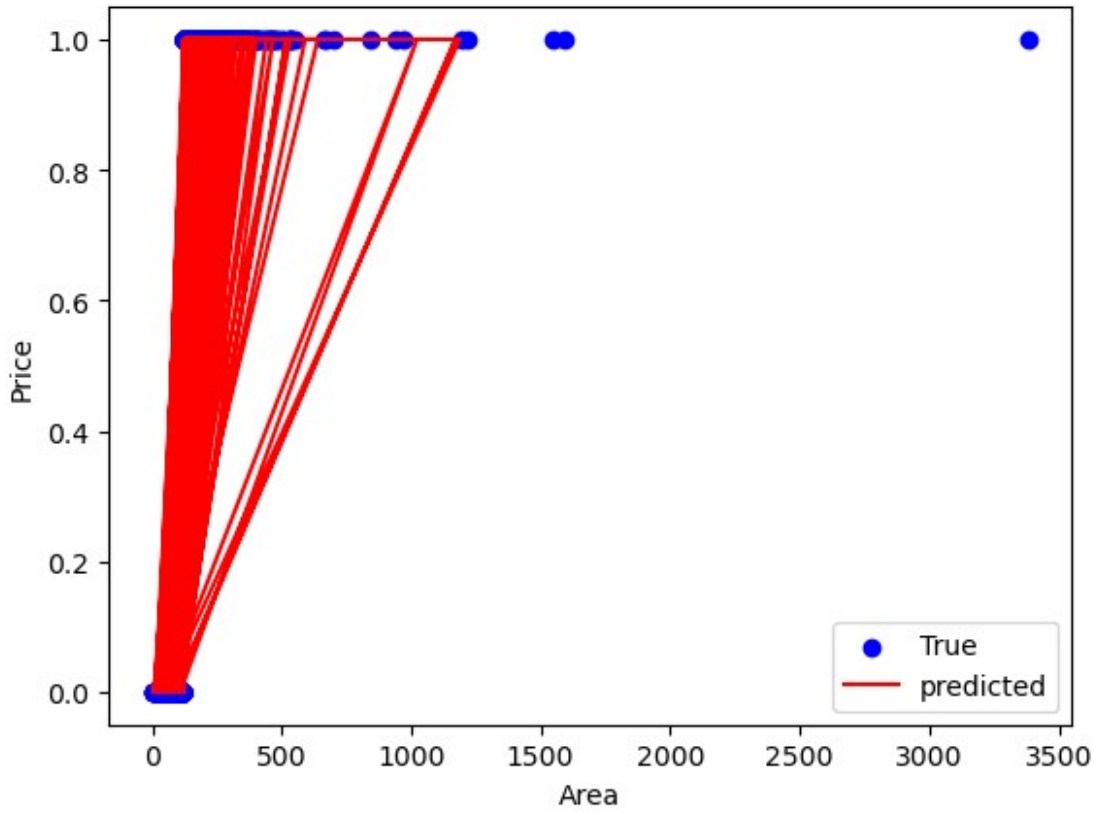
```
LogisticRegression()
```

```
model_LR.predict([[2900]]) #predicting
```

```
array([1])
```

```
predict_y = model_LR.predict(X_test)
```

```
plt.scatter(X_train,Y_train,c="blue",label="True")
plt.plot(X_test,predict_y,c="red",label="predicted")
plt.xlabel("Area")
plt.ylabel("Price")
plt.legend()
plt.show()
```




```
from sklearn.metrics import
confusion_matrix,classification_report,accuracy_score
```

```
cm = confusion_matrix(predict_y,Y_test)
```

```
cm
```

```
array([[428,   0],
       [  0, 322]])
```

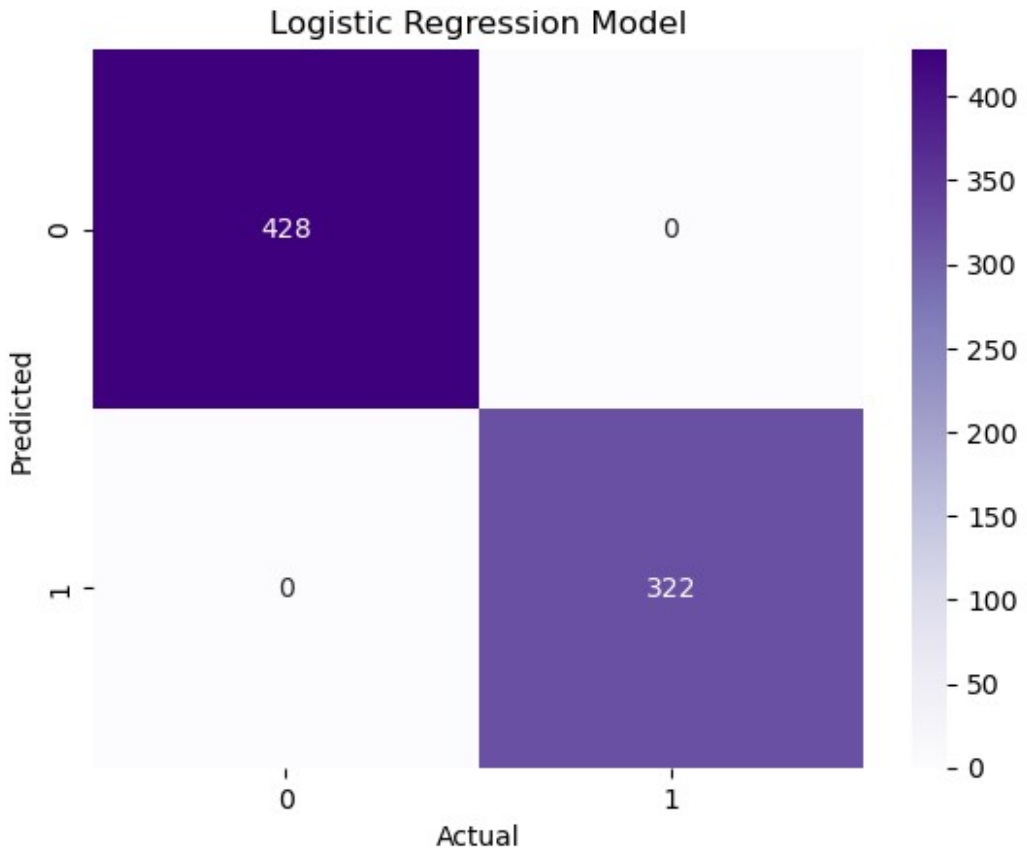
```
cr = classification_report(predict_y,Y_test)

print (cr)
```

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       428
           1       1.00      1.00      1.00       322

    accuracy                           1.00       750
   macro avg       1.00      1.00      1.00       750
weighted avg       1.00      1.00      1.00       750
```

```
acc = accuracy_score(predict_y,Y_test)

acc
```

```
1.0
```

```
import seaborn as sns

sns.heatmap(cm,cmap="Purples",annot=True,fmt=
".0f",xticklabels=[0,1],yticklabels=[0,1])
plt.xlabel("Actual")
plt.ylabel("Predicted")
plt.title("Logistic Regression Model")
plt.show()
```




```
sns.heatmap(cm,annot=True,cmap="Reds",fmt=".0f",xticklabels=[0,1],ytic
klabels=[0,1])
```

```
<Axes: >
```

5

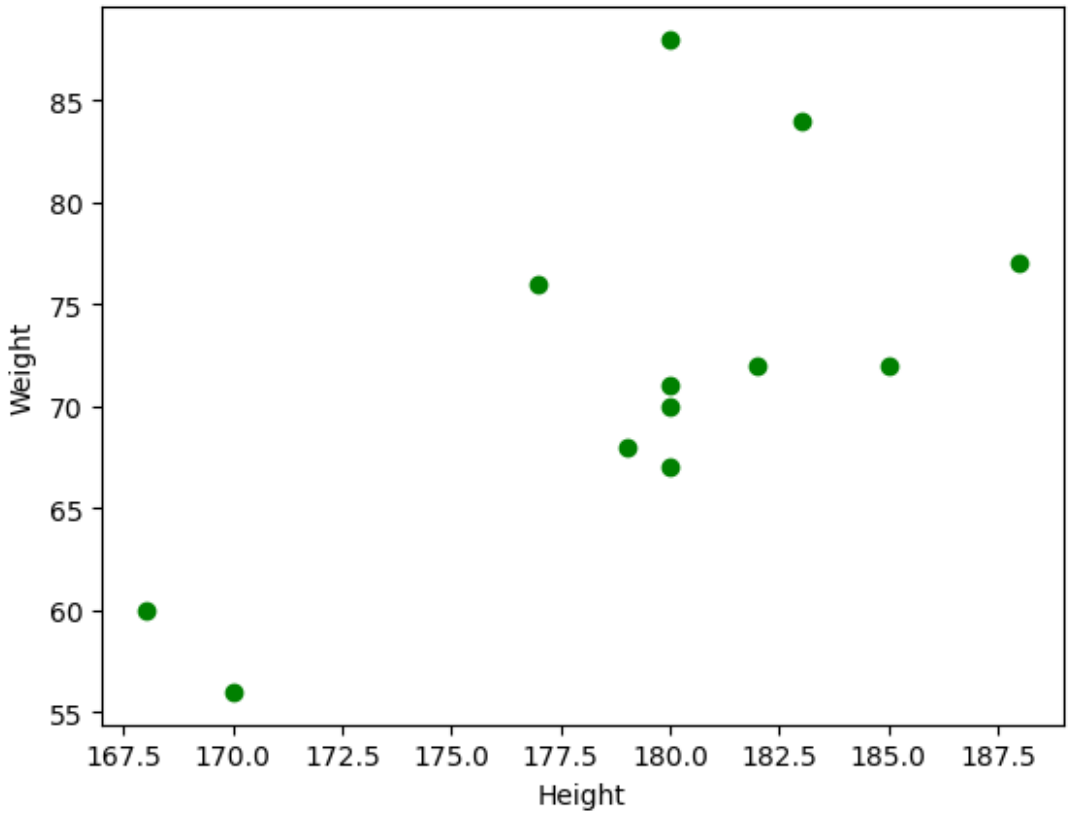
```
import numpy as np
import matplotlib.pyplot as plt

x = np.array([185,170,168,179,182,188,180,180,183,180,180,177]);

y = np.array([72,56,60,68,72,77,71,70,84,88,67,76])

print(len(x))
```

```
12
```

```
print(len(y))
```

```
12
```

```
plt.scatter(x,y,marker='o',c='green')
plt.xlabel("Height")
plt.ylabel("Weight")
plt.show()
```




```
from sklearn.cluster import KMeans

d = np.vstack((x,y)).T
kmean = KMeans(n_clusters=3,random_state=0)
```

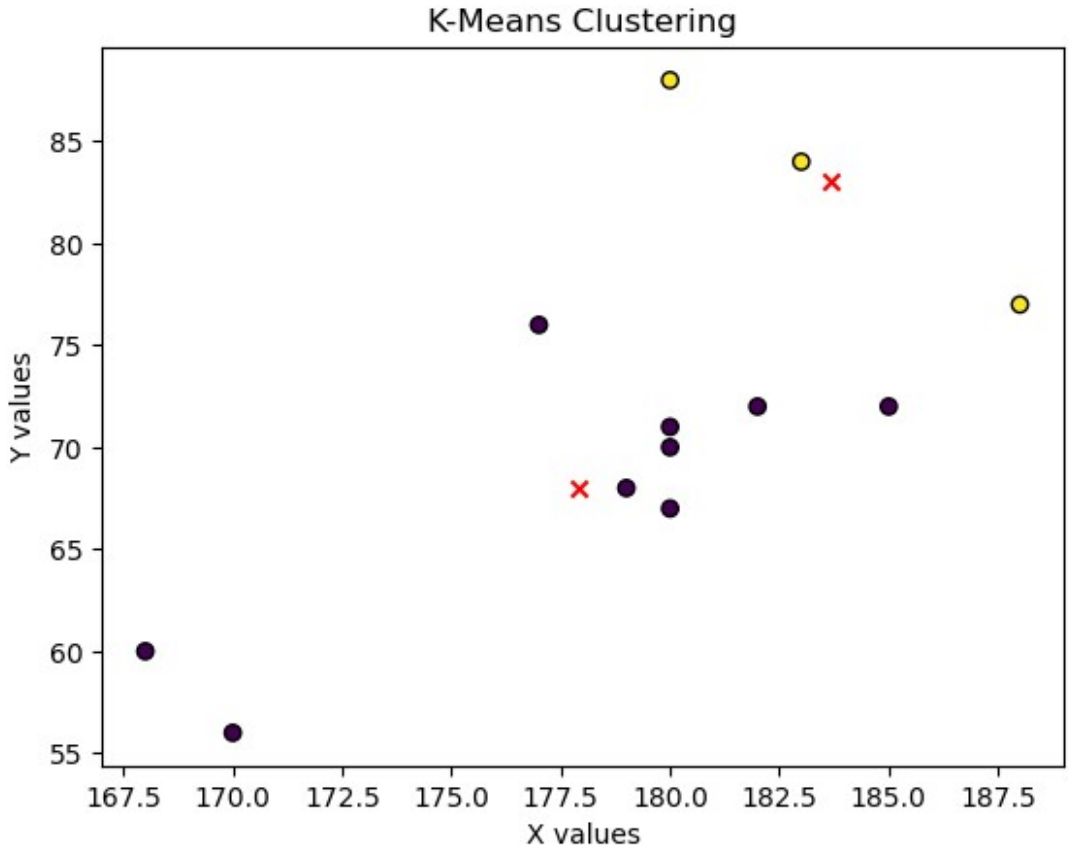
```
kmean.fit(d)
```

```
KMeans(n_clusters=3, random_state=0)
```

```
labels = kmeans.fit_predict(d)
```

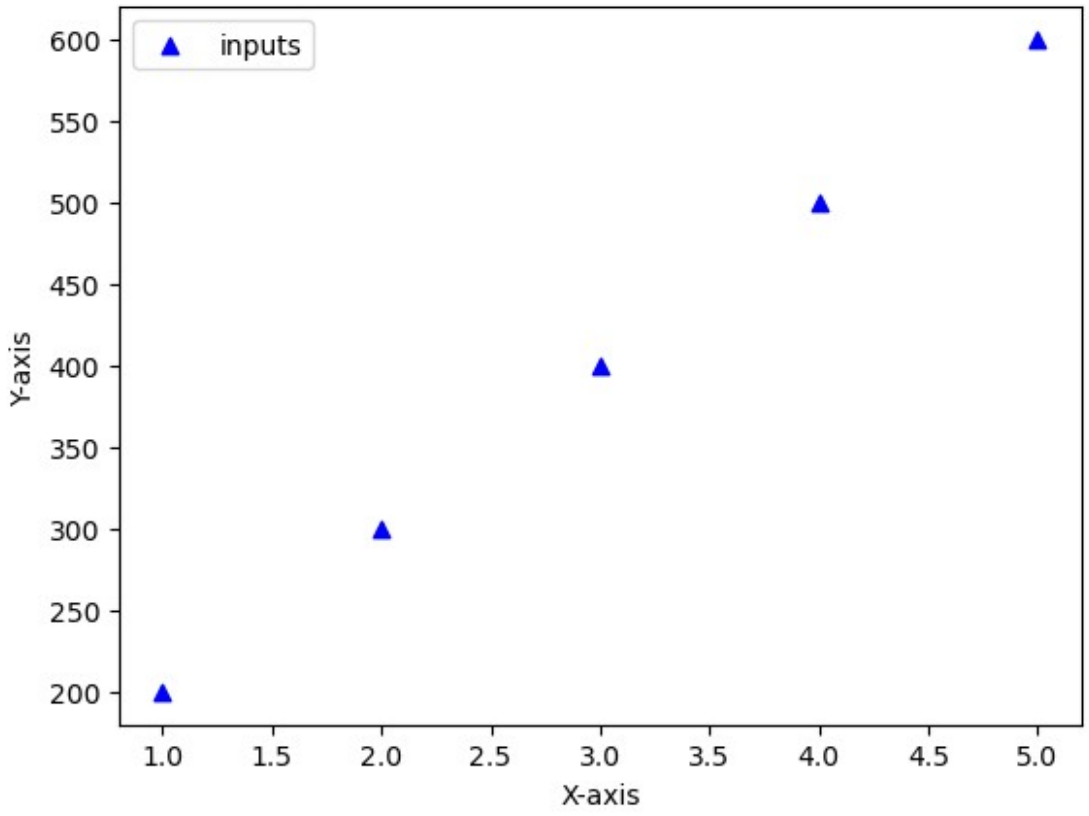
```
cen = kmeans.cluster_centers_
```

```
plt.scatter(d[:, 0], d[:, 1], c=labels, cmap='viridis', marker='o',
edgecolor='k')
plt.scatter(cen[:, 0], cen[:, 1], c='red', marker='x')
plt.title('K-Means Clustering')
plt.xlabel('X values')
plt.ylabel('Y values')
plt.show()
```



6.

```
import numpy as np
import matplotlib.pyplot as plt

X=np.array([1.0,2.0,3.0,4.0,5.0])
Y=np.array([200,300,400,500,600])

#create scatter plot
plt.scatter(X,Y,c='blue',marker='^',label='inputs')
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.legend()
plt.show()
```

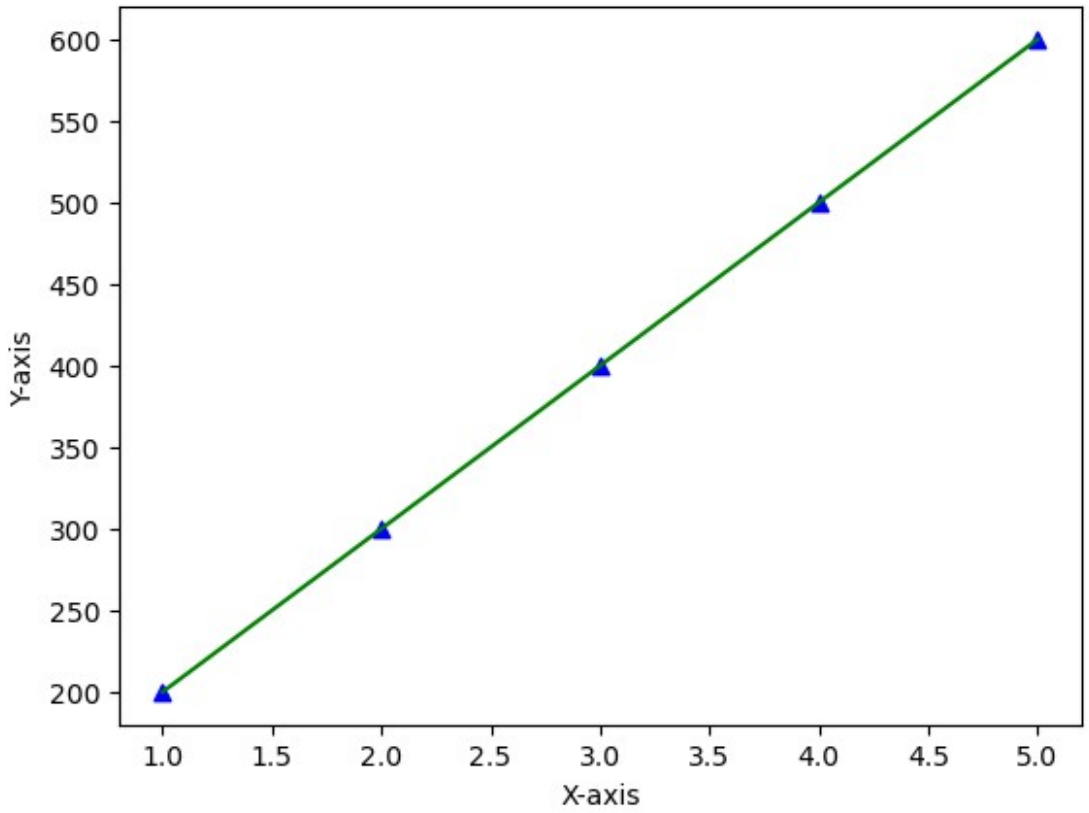



```
m=len(X)

#y^=f(x)=wx+b
def linear_regression_model(X,w,b):
    f_x = np.zeros(m)
    for i in range(m):
        f_x[i] = w*X[i]+b
    return f_x

w=100
b=100
```

```
predict_y = linear_regression_model(X,w,b)

print(predict_y)
```

```
[200. 300. 400. 500. 600.]
```

```
plt.scatter(X,Y,c='blue',marker='^',label='inputs')
plt.plot(X,predict_y,c='green')
plt.xlabel("X-axis")
plt.ylabel("Y-axis")
plt.show()
```




```
def cost_function_linear_regression(X, Y, w, b):
    m = len(X)
    sum_cost = 0

    for i in range(m):
        f_x = w * X[i] + b
        sum_cost += (f_x - Y[i]) ** 2

    final_cost = (1 / (2 * m)) * sum_cost
    return final_cost

j_cost_w_b = cost_function_linear_regression(X,Y,w,b)
```

```
print(f"minimum cost:{j_cost_w_b}")
```

```
minimum cost:0.0
```

```
def gradient_descent(X,Y,w,b):
    j_dw,j_db=0,0
    m = len(X)

    for i in range(m):
          f_x = w * X[i] + b
          j_dw += (f_x-Y[i])*X[i]
          j_db += (f_x-Y[i])

    j_dw=j_dw/m
    j_db=j_db/m
    return j_dw,j_db

j_dw,j_db = gradient_descent(X,Y,w,b)

print(f"j_dw:{j_dw} j_db:{j_dw}")
```

```
j_dw:0.0 j_db:0.0
```

```
def
gradient_descent_algo(X,Y,tempw,tempb,iteration,learningrate,gradient_
descent):
    w,b=tempw,tempb

    for i in range(iteration):
        j_dw,j_db=gradient_descent(X,Y,w,b)
        w = w - learningrate* j_dw
        b = b - learningrate* j_db
    return w,b


tempw=0
tempb=0
learningrate=0.001
iteration = 1000

w,b =
gradient_descent_algo(X,Y,tempw,tempb,iteration,learningrate,gradient_
descent)

for i in range(iteration):
 print(f"w:{w} b:{b}")
```

```
w:115.70560348553238 b:43.29466213948348
w:115.70560348553238 b:43.29466213948348
w:115.70560348553238 b:43.29466213948348
w:115.70560348553238 b:43.29466213948348
w:115.70560348553238 b:43.29466213948348
```

```
w:115.70560348553238 b:43.29466213948348
w:115.70560348553238 b:43.29466213948348
w:115.70560348553238 b:43.29466213948348
w:115.70560348553238 b:43.29466213948348
w:115.70560348553238 b:43.29466213948348
w:115.70560348553238 b:43.29466213948348
w:115.70560348553238 b:43.29466213948348
w:115.70560348553238 b:43.29466213948348
w:115.70560348553238 b:43.29466213948348
w:115.70560348553238 b:43.29466213948348
w:115.70560348553238 b:43.29466213948348
w:115.70560348553238 b:43.29466213948348
w:115.70560348553238 b:43.29466213948348
w:115.70560348553238 b:43.29466213948348
```

```
 print(f"w:{w} b:{b}")
```

```
w:115.70560348553238 b:43.29466213948348
```

In [3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

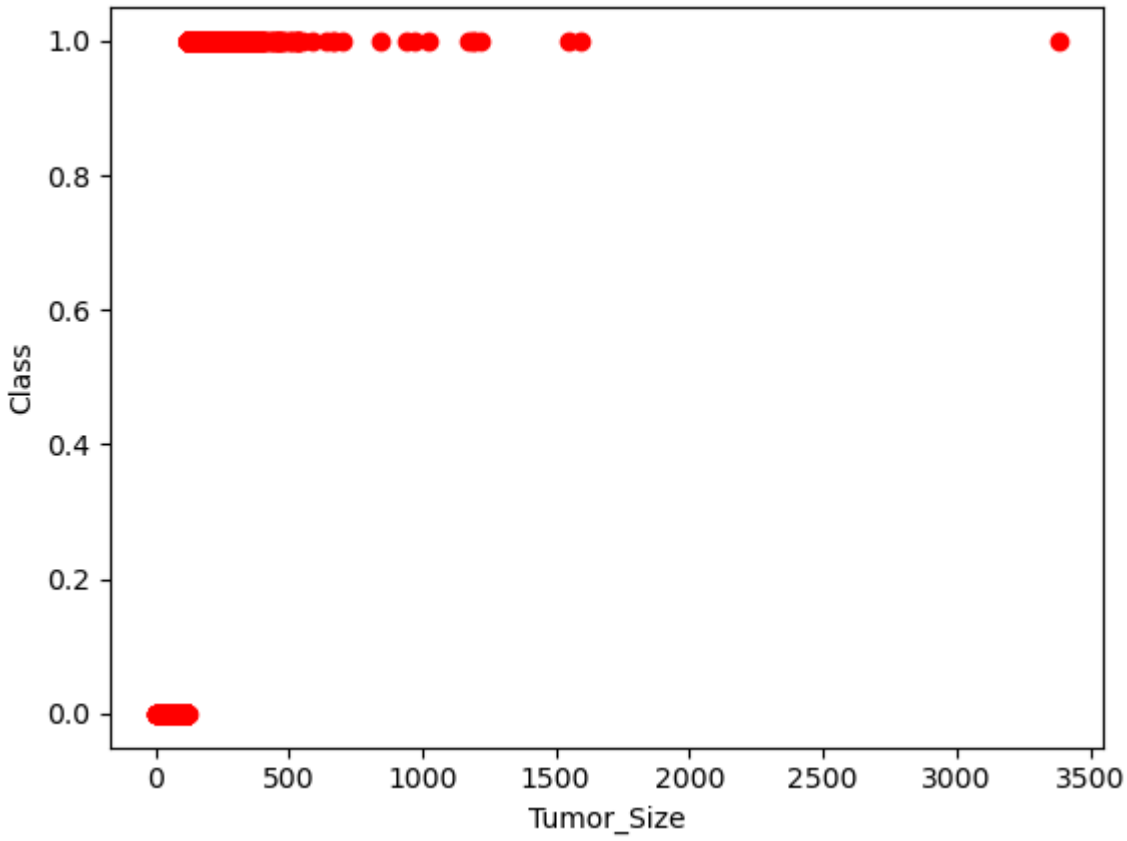
In [4]:
```
df = pd.read_csv("Brain_Tumor.csv")
df.head()
```

Out[4]:

	Class	Tumor_Size	Unnamed: 2
0	0	98.613971	NaN
1	0	63.858816	NaN
2	0	81.867206	s
3	1	151.229741	NaN
4	1	174.988756	NaN

In [5]:
```
X=df["Tumor_Size"]
Y=df["Class"]
```

In [6]:
```
plt.scatter(X,Y,marker='o',c='red')
plt.xlabel("Tumor_Size")
plt.ylabel("Class")
plt.show()
```



In [7]:
```
def sigmoid_func(x,w,b):
    f_x = np.dot(x,w)+b
    d = np.exp(-f_x) + 1
    return 1/d;
```

In [11]:
```
def cost_function(x,y,w,b):
    m = 1
    f_x = np.dot(x,w)+b
    pred = sigmoid_func(x,w,b)
    cost = (-1/m)*np.sum(y * np.log(pred)+(1-y)*np.log(1-pred))
    return cost
```

In [15]:
```
X = np.array(X).reshape(-1,1)
Y = np.array(Y).reshape(-1,1)
```

In [17]:
```
X_train = X[0:2623]
X_test = X[2623:]
Y_train = Y[0:2623]
Y_test = Y[2623:]
```

In [19]:
```
 w=0
 b=3
 x=X_test
 y=Y_test
 print(sigmoid_func(x,w,b))
 print(f"cost :{cost_function(x,y,w,b)}")
```

```
[[0.95257413]
 [0.95257413]
 [0.95257413]
 ...
 [0.95257413]
 [0.95257413]
 [0.95257413]]
cost :1824.6607705204615
```

In [ ]:
```

```