CLASS: B.E. E &TC **SUBJECT: ML**
EXPT. NO.: 2 **DATE:**
ROLL NO.: 42428

TITLE: Linear Regression using single neuron model

CO 1: Understand the basic concepts in machine learning like parametric/non- parametric modeling, classification, clustering, linear/ nonlinear regression and supervised/unsupervised learning to broadly classify various types of machine learning algorithms. Given an applicable feature vector, select an appropriate machine learning approach to design an analytical model to make expected predictions.

CO 5: Carry out experiments as an individual and in a team, comprehend and write a laboratory record and draw conclusions at a technical level.

AIM: To implement a simple regression technique using single neuron

SOFTWARES REQUIRED: MATLAB 7.0 or Python

THEORY:

1 Linear Regression

In data analytics we come across the term “Regression” very frequently. Before we continue to focus topic i.e. “Linear Regression” lets first know what we mean by Regression. Regression is a statistical way to establish a relationship between a dependent variable and a set of independent variable(s). e.g., if we say that

Age = 5 + Height * 10 + Weight * 13

Here we are establishing a relationship between Height & Weight of a person with his/ Her Age. This is a very basic example of Regression.

1.1 Introduction:

Least Square "Linear Regression" is a statistical method to regress the data with dependent variable having continuous values whereas independent variables can have either continuous or categorical values. In other words "Linear Regression" is a method to predict dependent variable (Y) based on values of independent variables (X). It can be used for the cases where we want to predict some continuous quantity. E.g., predicting traffic in a retail store.

1.2 Prerequisites:

To start with Linear Regression, you must be aware of a few basic concepts of statistics. i.e.

- Correlation (r) – Explains the relationship between two variables, possible values -1 to +1
- Variance (σ2) – Measure of spread in your data
- Standard Deviation (σ) – Measure of spread in your data (Square root of Variance)
- Normal distribution
- Residual (error term) – {Actual value – Predicted value}

1.3 Assumptions of Linear Regression

Not a single size fits or all, the same is true for Linear Regression as well. In order to fit a linear regression line data should satisfy few basic but important assumptions. If your data doesn't follow the assumptions, your results may be wrong as well as misleading.

i. Linearity & Additive:

There should be a linear relationship between dependent and independent variables and the impact of change in independent variable values should have additive impact on dependent variable.

ii. Normality of error distribution:

Distribution of differences between Actual & Predicted values (Residuals) should be normally distributed.

iii. Homoscedasticity:

Variance of errors should be constant versus,

a. Time

b. The predictions

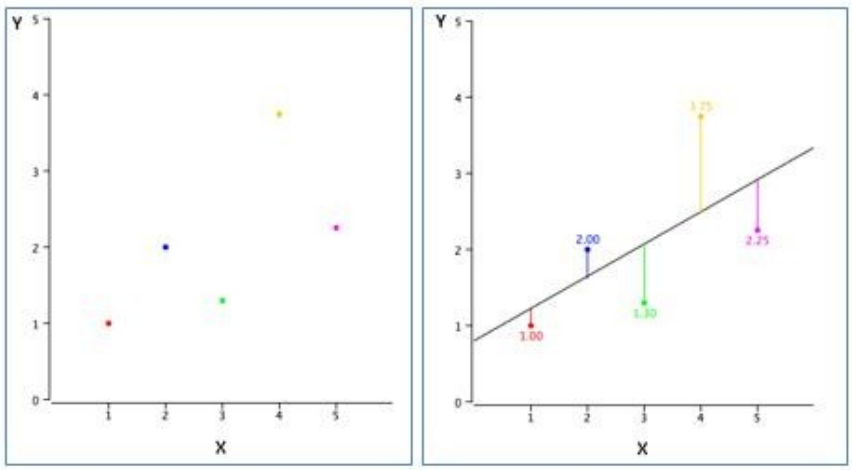
c. Independent variable values

iv. Statistical independence of errors:

The error terms (residuals) should not have any correlation among themselves. E.g., In case of time series data there shouldn't be any correlation between consecutive error terms.

v. Linear Regression Line:

While doing linear regression our objective is to fit a line through the distribution which is nearest to most of the points. Hence reducing the distance (error term) of data points from the fitted line. For example, in above figure (left) dots represent various data points and line (right) represents an approximate line which can explain the relationship between 'x' & 'y' axes.



Through, linear regression we try to find out such a line. For example, if we have one dependent variable 'Y' and one independent variable 'X' – relationship between 'X' & 'Y' can be represented in a form of following equation:

Y = B0 + B1X Where,

- Y = Dependent Variable
- X = Independent Variable
- B0 = Constant term a.k.a Intercept
- B1 = Coefficient of relationship between 'X' & 'Y'

1.3.1 Few properties of linear regression line

- Regression line always passes through mean of independent variable (x) as well as mean of dependent variable (y).

- Regression line minimizes the sum of “Square of Residuals”. That’s why the method of Linear Regression is known as “Ordinary Least Square (OLS)” Why to reduce “Square of errors” and not just the errors?
- B1 explains the change in Y with a change in X by one unit. In other words, if we increase the value of ‘X’ by one unit then what will be the change in value of Y Will correlation coefficient between ‘X’ and ‘Y’ be same as B1?

1.4 Finding a Linear Regression Line

Using a statistical tool e.g., Excel, R, SAS etc. you will directly find constants (B0and B1) as a result of linear regression function. But conceptually as discussed it works on OLS concept and tries to reduce the square of errors, using the very concept software packages calculate these constants.

For example, let say we want to predict ‘y’ from ‘x’ given in following table and let’s assume that our regression equation will look like

“y=B0+B1*x”

X	y	Predicted 'y'
1	2	B0+B1*1
2	1	B0+B1*2
3	3	B0+B1*3
4	6	B0+B1*4
5	9	B0+B1*5
6	11	B0+B1*6
7	13	B0+B1*7
8	15	B0+B1*8

9	17	B0+B1*9
10	20	B0+B1*10

Where,

Std. Dev. of x	3.02765
Std. Dev. of y	6.617317
Mean of x	5.5
Mean of y	9.7
Correlation between x & y	.989938

If we differentiate the Residual Sum of Square (RSS) w.r.t B0 & B1 equate the results to zero, we get the following equations as a result:

B1 = Correlation * (Std. Dev. of y/ Std. Dev. of x)

B0 = Mean(Y) – B1 * Mean(X)

Putting values from table 1 into the above equations,

B1 = 2.64

B0 = -2.2

Hence, the least regression equation will become

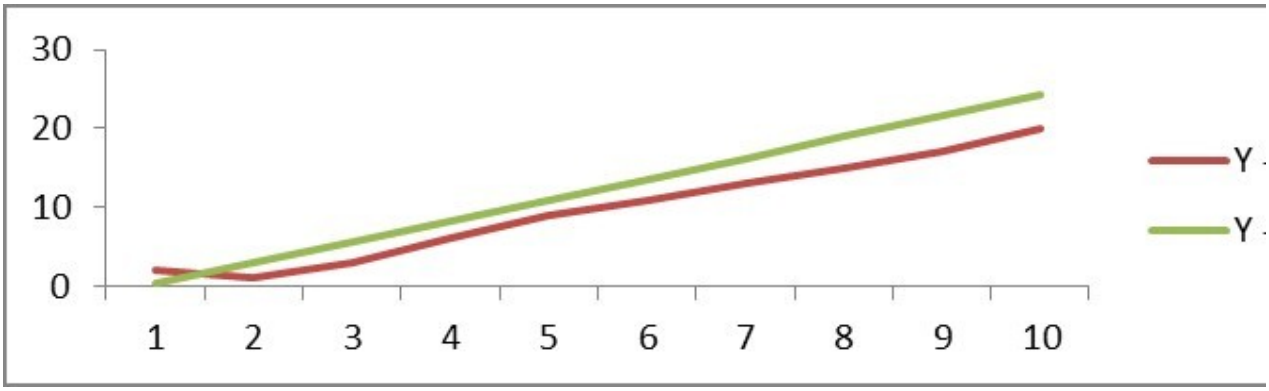
Y = -2.2 + 2.64*x

Let see, how our predictions are looking like using this equation

X	Y - Actual	Y - Predicted
1	2	0.44
2	1	3.08
3	3	5.72
4	6	8.36
5	9	11

6	11	13.64
7	13	16.28
8	15	18.92
9	17	21.56
10	20	24.2

Given only 10 data points to fit a line our predictions are not pretty accurate but if we see the correlation between 'Y-Actual' & 'Y – Predicted' it will turn out to be very high; hence both the series are moving together and here is the graph for visualizing our prediction values:



1.5 Model Performance:

Once you build the model, the next logical question comes in mind is to know whether your model is good enough to predict in future or the relationship which you built between dependent and independent variables is good enough or not.

For this purpose, there are various metrics which we look into-R – Square (R2)

Formula for calculating R2 is given by:

$$R^2 = \frac{TSS - RSS}{TSS}$$

1.6 Problem on Linear Regression:

The evaluation of 10 students is carried out on the basis of correct answers and the corresponding aptitude. Find the linear regression function.

Correct Answers (x)	Aptitude(y)
17	94
13	73
12	69
15	80
16	93
14	85
16	66
16	79
18	77
19	91

Linear Regression Function:

$$Y = a + bX$$

Where, a is intercept and b is slope of the line.

Slope (b) of the regression line:

$$b = r.\ (Sy/Sx)$$

X	Y			((	$(X-X`)^2$	
17	94	1.4	13.3	20.02	1.96	204.49
13	73	-2.6	-7.7	17.42	6.76	44.39
12	59	-3.6	-11.7	74.52	12.96	428.49
15	80	-0.6	0.7	-0.18	0.36	0.09
16	93	0.4	12.3	-5.32	0.16	176.89
14	85	-1.6	4.3	-8.48	2.56	28.09
16	66	0.4	-14.7	-5.48	0.16	187.69
16	79	0.4	-1.7	-0.28	0.16	0.49
18	77	2.4	-3.7	-6.48	5.76	7.29
19	91	3.4	10.3	38.42	11.56	127.69
=15.6	=80.7			=134.8	=42.4	=1206.1

Correlation Coefficient (r):

r =
r = 134.8/ (42.4*1206.1) ^0.5
r = 0.596
Here n is number of students
Sy = ^0.5
Sy = (1206.1/9)^0.5
Sy = 9.9

Sx = ^0.5
Sx = (42.4/9)^0.5
Sx = 2.171

b = r*(Sy/Sx)
b = 0.596*9.9/2.171
b = 2.326
a =
a = 80.7 – 2.326*15.6
a = 44.41

Y = a + bX
Hence the regression function is ,
Y = 44.41 + 2.326 *X

For X=15,
Y =44.41 + (2.326*15)
Y = 79.3

CONCLUSION:

The artificial equivalent of a neuron is a node (also sometimes called neurons, but refer to them as nodes to avoid ambiguity) that receives a set of weighted inputs, processes their sum with its activation function φφ, and passes the result of the activation function to nodes further down the graph. In fact, the simplest neural network performs least squares regression. We

will then use gradient descent on the loss's gradient ∇wL(w)∇wL(w) in order to minimize the overall error on the training data.

REFERENCES:

i. Laurene Fausett, “Fundamentals of Neural Networks: Architectures, Algorithms And Applications”, Pearson Education, Inc, 2008.

ii. Winston, Patrick Henry. “Artificial Intelligence” 3rd Edition, Addison-Wesley, 1992.

iii. Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, “ImageNet Classification with Deep Convolutional Neural Networks”, University of Toront

(Course Teacher)

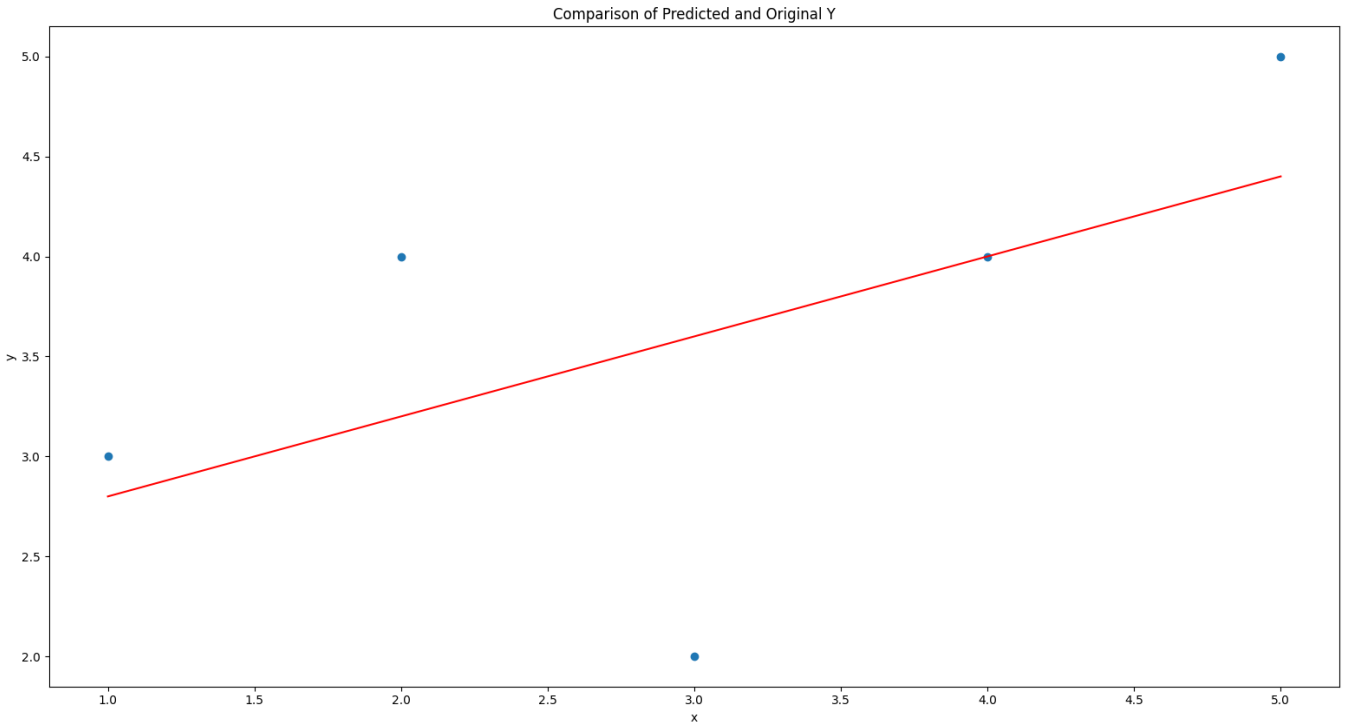
Code :

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

x = np.array([1,2,3,4,5])
y = np.array([3,4,2,4,5])

mean_x = np.mean(x)
mean_y = np.mean(y)

numerator = 0
denominator = 0

for i in range(x.size):
    numerator += (x[i] - mean_x)*(y[i] - mean_y)
    denominator += (x[i] - mean_x)**2

m = numerator/denominator

c = mean_y - (m * mean_x)

print("\nsolpe = ",m)
print("\nintercept = ",c)
print("\nLine fitted : y =",m,"x +",c)

y_predicted = np.zeros(y.size)

for i in range(x.size):
    y_predicted[i] = (m * x[i]) + c

print("\nx = ",x)
print("\ny = ",y)
print("\ny_predicted = ",y_predicted)

numerator = 0
denominator = 0

for i in range(x.size):
    numerator += (y_predicted[i] - y[i])**2
    denominator += (y[i] - mean_y)**2
```

```
R_sq_coeff = numerator/denominator

print("\nR_square Coefficient = ",R_sq_coeff)

plt.scatter(x,y)
plt.plot(x,y_predicted,'r')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Comparison of Predicted and Original Y')
plt.show()
```

Output :

```
solpe =  0.4

intercept =  2.4

Line fitted : y = 0.4 x + 2.4

x =  [1 2 3 4 5]

y =  [3 4 2 4 5]

y_predicted =  [2.8 3.2 3.6 4.  4.4]

R_square Coefficient =  0.6923076923076922
PS C:\Users\chand>
```



Code :

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import linear_model

df = pd.read_csv('areas.csv')

plt.xlabel('area')
plt.ylabel('prices')
plt.scatter(df.area,df.price,color='red',marker='*')
plt.show()

x = df.iloc[:,0:1].values
x = np.array(df.area)
y = df.price
x = df.drop(['price'],axis=1)
```

```
model=linear_model.LinearRegression()

#model.fit([df.area],[df.price])
#model.fit(x,y)

new_area=df.drop('price',axis='columns')
#print(new_area)
model.fit(new_area,df.price)

print('m = ',model.coef_,' and c = ',model.intercept_)

print('For area = 6000 predicted price is ',model.predict([[6000]]))
print('For area = 6000 calculated (y=mx+c) price is ',(6000*model.coef_)+model.intercept
_)
```

Dataset :

area,price

1000,200000

1500,290000

2300,320000

3540,350000

4120,430000

4560,480000

5490,530000

3460,550000

4750,650000

2300,750000

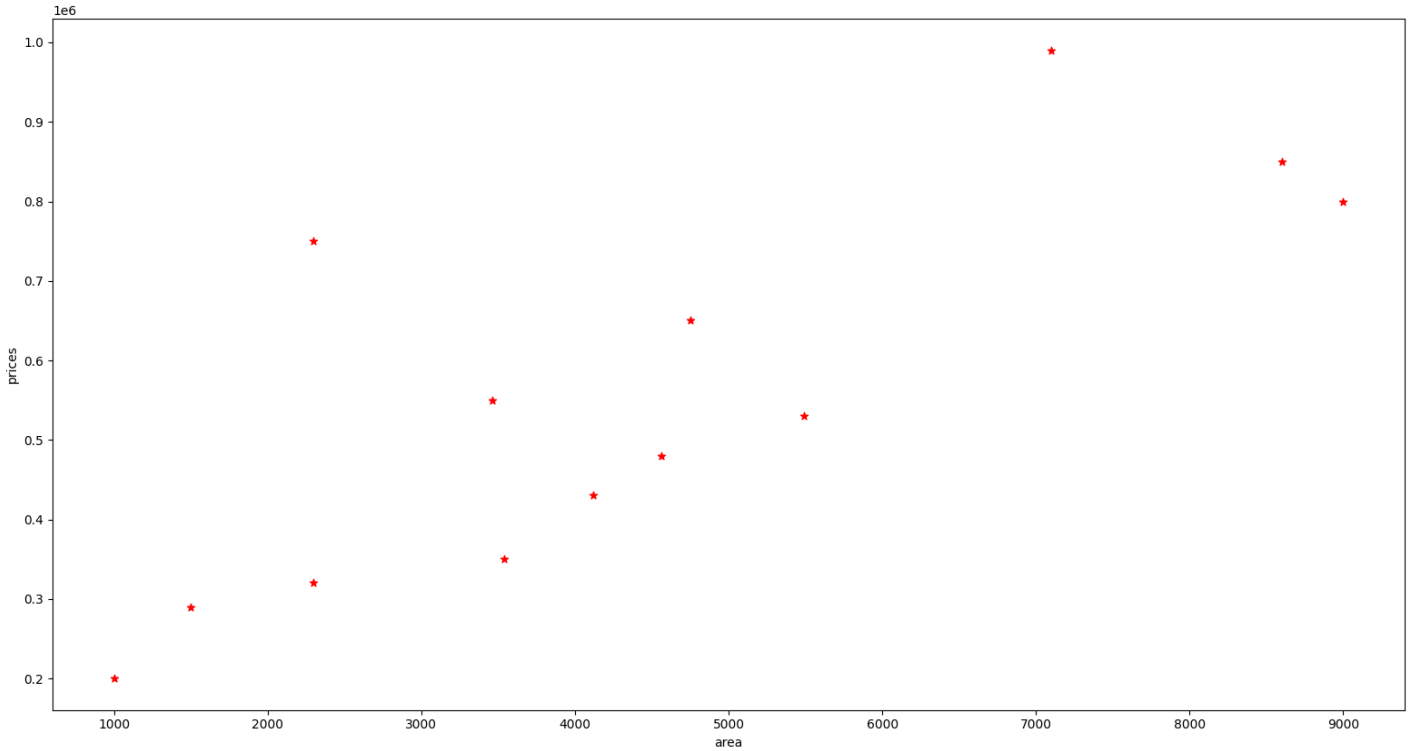
9000,800000

8600,850000

7100,990000

Output :

```
m =  [74.59850829]  and c =  221859.54628449207
For area = 6000 predicted price is  [669450.59600399]
For area = 6000 calculated (y=mx+c) price is  [669450.59600399]
```



Code :

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import linear_model
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

df = pd.read_csv('data.csv')

plt.xlabel('x')
plt.ylabel('y')
plt.scatter(df.x,df.y,color='red',marker='*')
plt.show()
```

```
x = df.iloc[:,0:1].values
x = np.array(df.x)
y = df.y
x = df.drop(['y'],axis=1)

#use this number of samples is huge
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=0)

lireg=linear_model.LinearRegression()

lireg.fit(x_train,y_train)

plt.xlabel('x_train')
plt.ylabel('y_train')
plt.scatter(x_train,y_train,color='blue',marker='*')
plt.show()

print('m = ',lireg.coef_,' and c = ',lireg.intercept_)

y_train_pred = lireg.predict(x_train)
y_test_pred = lireg.predict(x_test)

print('Training M.S.V is ',mean_squared_error(y_train,y_train_pred))
print('Testing M.S.V is ',mean_squared_error(y_test,y_test_pred))
```

Dataset :

x,y

10,95

9,80

2,10

15,50

10,45

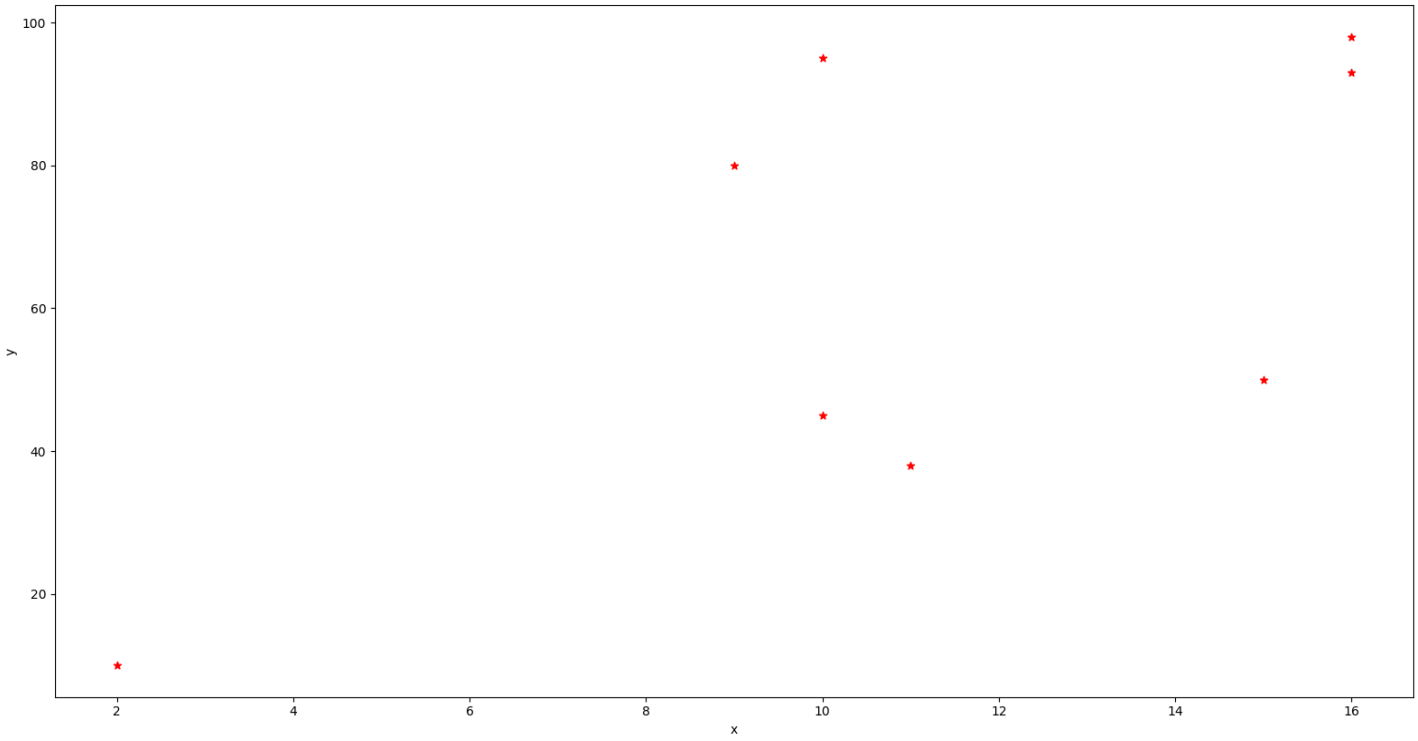
16,98

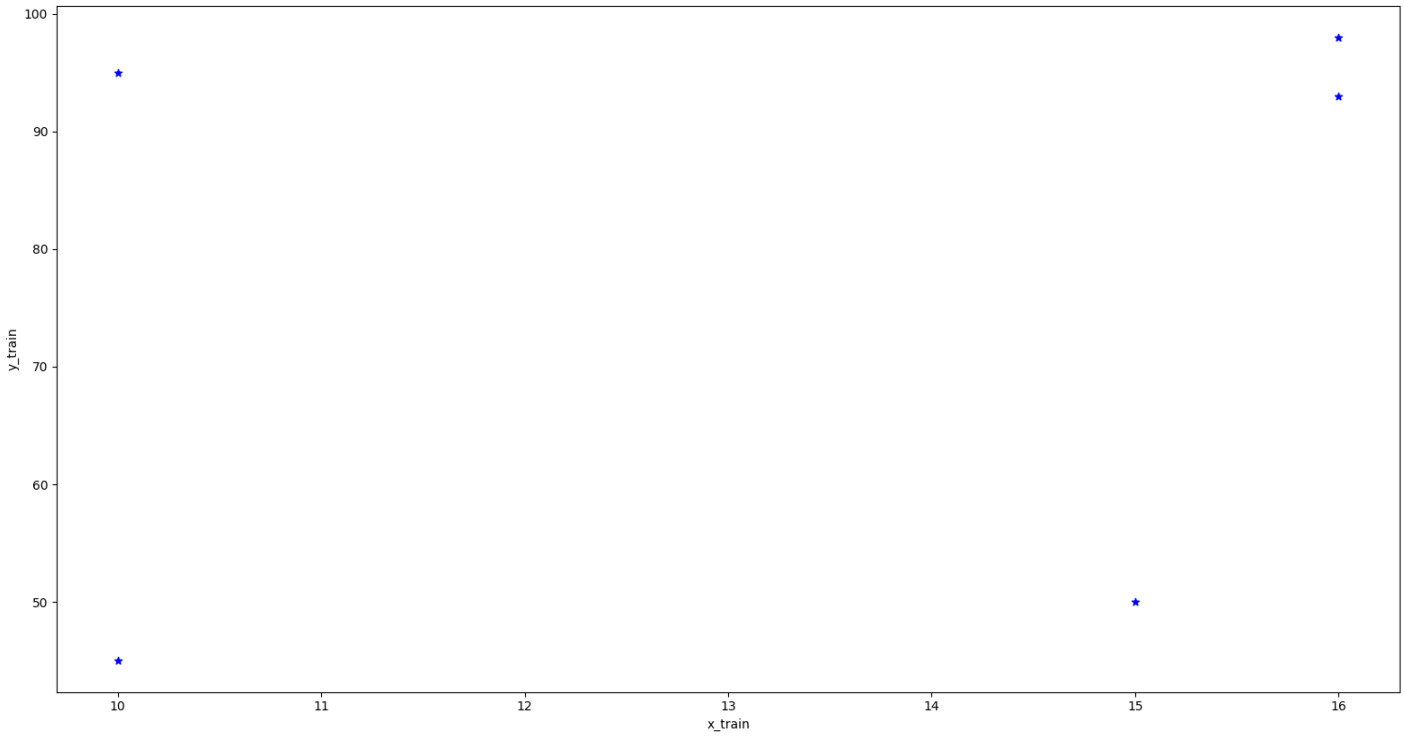
11,38

16,93

Output :

```
m =  [2.56632653]  and c =  41.81122448979591
Training M.S.V is  502.5255102040818
Testing M.S.V is  873.0758451339722
```




100
90
80
70
60
50
y_train
10
11
12
13
14
15
16
x_train