NAME :- HAMMAD ABID (9134)

x	6.11	5.52	8.51	7.00
y	17.59	9.13	13.66	11.85

$\theta = [0,0]$

$h\theta(x) = \theta_0 + \theta_1 x$ [redacted]

$h\theta^{(1)} = 0 + 0(6.11)$

$\boxed{h\theta x^{(1)} = 0}$

$h\theta x^{(2)} = 0 + 0(5.52)$

$\boxed{h\theta x^{(2)} = 0}$

$h\theta x^{(3)} = 0 + 0(8.51)$

$\boxed{h\theta x^{(3)} = 0}$

$h\theta(x^{(4)} = 0 + 0(7.00)$

$\boxed{h\theta x^{(4)} = 0}$

Formula for gradient Decent:

$$\theta_j = \theta_j - \frac{\alpha}{m}\sum_{i=1}^{m}\left[h\theta(x_i) - y_i)x_i\right]$$

$$\underbrace{\begin{array}{l} h\theta(x1) - y_1 = 0 - 17.59 = -17.59 \\ h\theta(x^2) - y_2 = 0 - 9.13 = -9.13 \\ h\theta(x^3) - y_3 = 0 - 13.66 = -13.66 \\ h\theta(x^4) - y_4 = 0 - 11.85 = -11.85 \end{array}}_{\theta_0}$$

$h\theta(x_1) - y_1 * x_1 = -17.59 * 6.11 = -107.4749$

$h\theta(x_2) - y_2 * x_2 = -9.13 * 5.52 = -50.3976$

$h\theta(x_3) - y_3 * x_3 = -13.66 * 8.51 = -116.2466$

$h\theta(x^4) - y_4 * x_4 = -11.85 * 7.00 = -82.95$

$\theta_0 = \theta_0 - \frac{\alpha}{4}\sum_{i=1}(h\theta(x^1) - y^1)$

$= 0 - \frac{0.01}{4}((-17.59) + (-9.13) + (-13.66) + (-11.85))$

$$= 0 - \frac{0.01}{4}(-52.23)$$

$$\theta_0 = 0 - - 0.130575 = \boxed{0.130575}$$

$$\theta_1 = \theta_1 - \frac{\alpha}{m}(h\theta(x^1) - y) * x$$

$$\theta_1 = 0 - \frac{0.01}{4}(-107.4749 + (-50.3976) \; (-116.2466) + (-82.95)$$

$$\boxed{\theta_1 = 0.892672}$$

Second Iteration:. $\theta = \begin{array}{cc} \theta_0 & \theta_1 \\ [0.13, & 0.89] \end{array}$

$h\theta(x^{(1)}) - y^{(1)} = 5.58 - 17.59 = -12.00$

$h\theta(y^{(2)}) - y(2) = 5.05 - 9.13 = -4.07$

$h\theta(x^{(3)}) - y(3) = 7.72 - 13.66 = -5.93$

$h\theta(x^{(4)}) - y^{(4)} = 6.37 - 11.85 = -5.47$

$(h\theta(x^{(1)}) - y^{(1)}) *$ ~~$x^{(1)}$~~ $=$ ~~$-12 *$~~ $5.58 - 17.59 = -12.0221$

$(h\theta(x^{(2)}) - y_2) = 5.0428 - 9.13 = -4.0872$

$(h\theta(x^{(3)}) - y_3) = 7.7039 - 13.66 = -5.9561$

$(h\theta(x^{(4)}) - y_4) = 6.33 - 11.85 = -5.52$

$h\theta(x_1) - y_1 * x_1 = -12.0221 * 6.11 = -73.45$

$h\theta(x_2) - y_2 * x_2 = -4.0872 * 5.52 = -22.56$

$h\theta(x_3) - y_3 * x_3 = -5.9561 * 8.51 = -50.68$

$h\theta(x_4) - y_4 * x_4 = -5.52 * 7.00 = -38.64$

$$\theta_0 = \theta_0 - \frac{\alpha}{m}\sum_{i=1}^{4}(h\theta(x^{(i)}) - y)$$

$$= 0.13 - \frac{0.01}{4}\sum_{i=1}^{4}(-12.0221 - 4.0872 - 5.4561 - 5.5$$

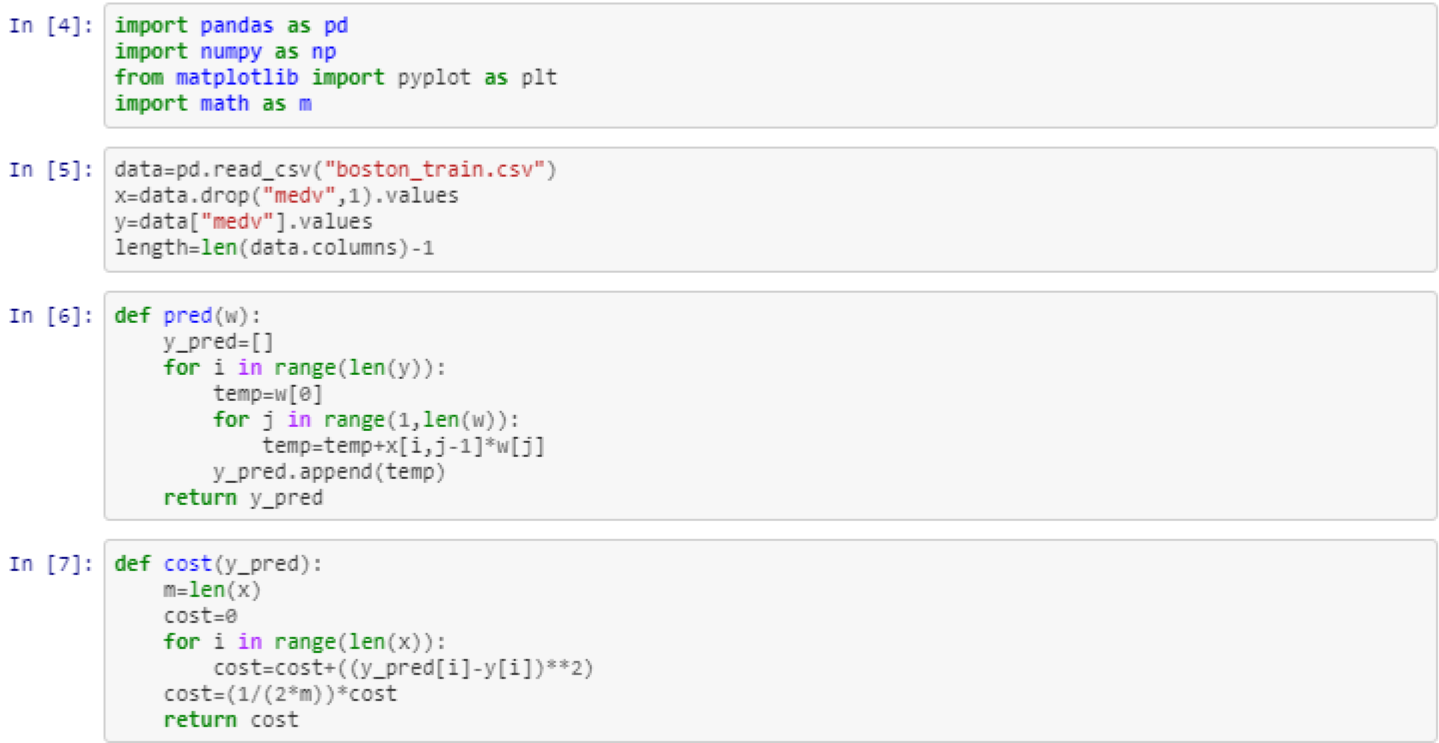
$$\theta_0 = 0.13 - (-0.06) = \boxed{0.19}$$

$$\theta_1 = 0.89 - \frac{0.01}{4}\sum_{i=1}^{4}(-73.45 - 22.56 - 50.68 - 38.64)$$

$$\theta_1 = 0.89 - (-0.46)$$

$$\boxed{\theta_1 = 1.35} \Rightarrow \theta = \begin{bmatrix}\theta_0 \\ \theta_1\end{bmatrix} = \begin{bmatrix}0.19 \\ 1.35\end{bmatrix}$$

Python code for cost function:



In [4]:

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
import math as m
```

In [5]:

```
data=pd.read_csv("boston_train.csv")
x=data.drop("medv",1).values
y=data["medv"].values
length=len(data.columns)-1
```

In [6]:

```
def pred(w):
    y_pred=[]
    for i in range(len(y)):
        temp=w[0]
        for j in range(1,len(w)):
            temp=temp+x[i,j-1]*w[j]
        y_pred.append(temp)
    return y_pred
```

In [7]:

```
def cost(y_pred):
    m=len(x)
    cost=0
    for i in range(len(x)):
        cost=cost+((y_pred[i]-y[i])**2)
    cost=(1/(2*m))*cost
    return cost
```

Question 3ii(b)

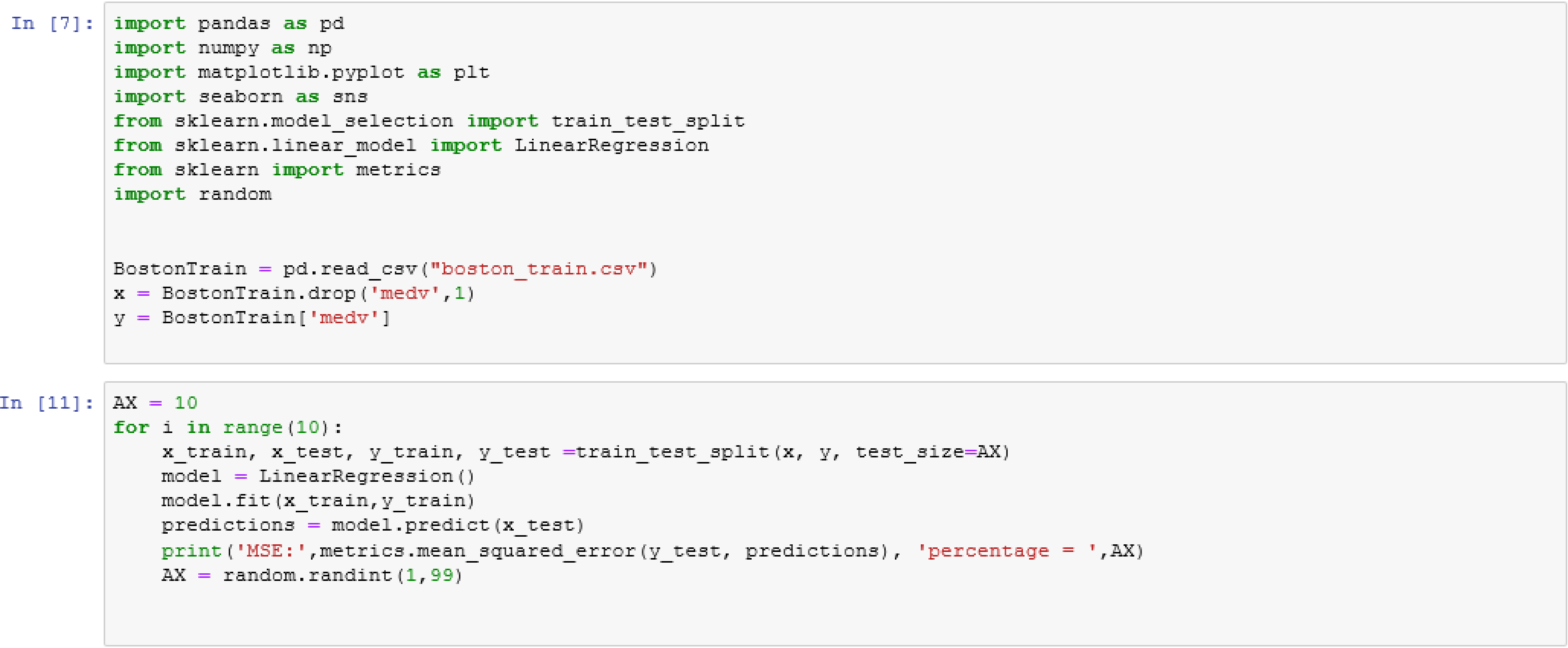
```
# Boston Housing with Linear Regression
# With this data our objective is create a
model using linear regression to predict the
houses price
#
# The data contains the following columns:
#
# 'crim': per capita crime rate by town.
# 'zn': proportion of residential land zoned
for lots over 25,000 sq.ft.
# 'indus': proportion of non-retail business
acres per town.
# 'chas':Charles River dummy variable (= 1 if
tract bounds river; 0 otherwise).
# 'nox': nitrogen oxides concentration (parts
per 10 million).
# 'rm': average number of rooms per dwelling.
# 'age': proportion of owner-occupied units
built prior to 1940.
# 'dis': weighted mean of distances to five
Boston employment centres.
# 'rad': index of accessibility to radial
highways.
# 'tax': full-value property-tax rate per
$10,000.
# 'ptratio': pupil-teacher ratio by town
# 'black': 1000(Bk - 0.63)^2 where Bk is the
proportion of blacks by town.
# 'lstat': lower status of the population
(percent).
# 'medv': median value of owner-occupied homes
in $$1000s

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

```
train_test_split
from sklearn.linear_model import
LinearRegression,Lasso,Ridge

# pd.set_option('display.max_columns',1000)
# pd.set_option('display.width',1000)
# np.random.seed(2)
# Importing DataSet and take a look at Data

BostonTrain = pd.read_csv("boston_train.csv")
# ID columns does not relevant for our
analysis.
BostonTrain.drop('ID', axis = 1, inplace=True)

X = BostonTrain[['crim', 'zn', 'indus', 'chas',
'nox', 'rm', 'age', 'dis', 'rad', 'tax',
       'ptratio', 'black', 'lstat']]
y = BostonTrain['medv']


X_train, X_test, y_train, y_test =
train_test_split(X, y, test_size=0.4)

model = LinearRegression()
model.fit(X_train,y_train)

predictions = model.predict(X_test)

print(predictions)

plt.scatter(y_test, predictions)
plt.xlabel('Y Test')
plt.ylabel('Predicted Y')
plt.show()

from sklearn import metrics
print('simple Linear Regression')
print('MAE:',
metrics.mean_absolute_error(y_test,
predictions))
print('MSE:',
metrics.mean_squared_error(y_test,
predictions))
print('RMSE:',
np.sqrt(metrics.mean_squared_error(y_test,
predictions)))
```

ANALYZATION:

Not possible to make a linear line because it has many feature.

Question 3ii(c)

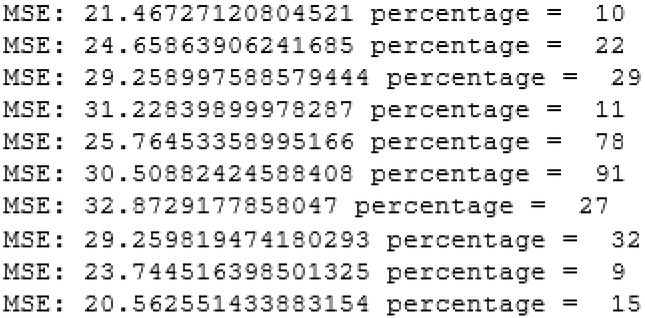


In [7]:

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


BostonTrain = pd.read_csv("boston_train.csv")
x = BostonTrain.drop('medv',1)
y = BostonTrain['medv']
```

In [11]:

```
AX = 10
for i in range(10):
    x_train, x_test, y_train, y_test =train_test_split(x, y, test_size=AX)
    model = LinearRegression()
    model.fit(x_train,y_train)
    predictions = model.predict(x_test)
    print('MSE:',metrics.mean_squared_error(y_test, predictions), 'percentage = ',AX)
    AX = random.randint(1,99)
```



```
MSE: 21.46727120804521 percentage =  10
MSE: 24.65863906241685 percentage =  22
MSE: 29.258997588579444 percentage =  29
MSE: 31.22839899978287 percentage =  11
MSE: 25.76453358995166 percentage =  78
MSE: 30.50882424588408 percentage =  91
MSE: 32.8729177858047 percentage =  27
MSE: 29.259819474180293 percentage =  32
MSE: 23.744516398501325 percentage =  9
MSE: 20.562551433883154 percentage =  15
```

MY OBSERVATION: % jitna kam uthana acha result

Question 2

X_1	4.85	8.62	5.46	9.21
X_2	9.63	3.23	8.23	6.34
y	1	0	1	0

$$\theta_1 = \theta_j \alpha \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) * x^{(i)}$$

$\theta's = [0,0,0]$

$z_1 = z_2 = z_3 = 0$

Now cal hypothesis

$$h_\theta(x^{(1)}) = \frac{1}{1+e^{-0}} = \frac{1}{2} = 0.5$$

$$h_\theta(x^{(2)}) = \frac{1}{1+e^{-0}} = \frac{1}{2} = 0.5$$

$$h_\theta(x^{(3)}) = \frac{1}{1+e^{-0}} = \frac{1}{2} = 0.5$$

$$h_\theta(x^{(4)}) = \frac{1}{1+e^{-0}} = \frac{1}{2} = 0.5$$

$$\theta_0 = 0 - 0.1 \{(0.5 - 1) + (0.5 - 0) + (0$$
$$+ (0.5 - 0)\}$$

$$\boxed{\theta_0 = 0}$$

$$\theta_1 = 0 - 0.01\{(0.5-1) * 4.85 + (0.5-0) * 8.62$$
$$5.46 + (0.5-0) * 9.21\}$$

$$\theta_1 = -0.0376$$

$$\theta_2 = 0.01 \{(0.5-1) * 9.63 + (0.5-0) * 3.23$$
$$(0.5-1) * 8.23 + (0.5-0) * 6.34$$

$$\boxed{\theta_2 = 0.041}$$

Question 2ii(b)

```python
# import numpy as np
import pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from matplotlib import pyplot as plt
# matplotlib inline

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

print(data.shape)
print(data)

age = data.drop('buy', axis=1)

train_x = age[:17]
test_x = age[-10:]

train_y = data.buy[:17]
test_y = data.buy[-10:]

plt.scatter(data.age,data.buy,marker='+',color='red')

model = LogisticRegression()
model.fit(train_x, train_y)

y_predicted = model.predict(test_x)
print(y_predicted)

train_Acc = model.score(train_x, train_y)
print(train_Acc)

print(test_y)
test_Acc = accuracy_score(test_y, y_predicted)
print(test_Acc)
```