12/27/2022

ML

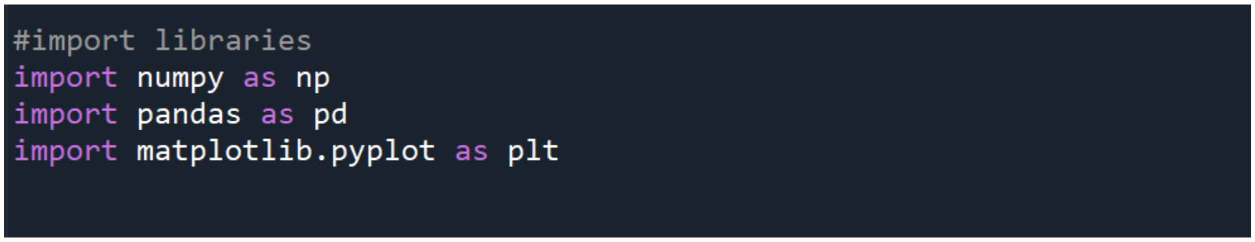
Regression

TEAM MEMBERS:

1. HOSSAM MOHAMMED ALI
2. KHALED AHMED ISMAIL

Dr. Kareem Ezz El-Deen

FCI FAYOUM



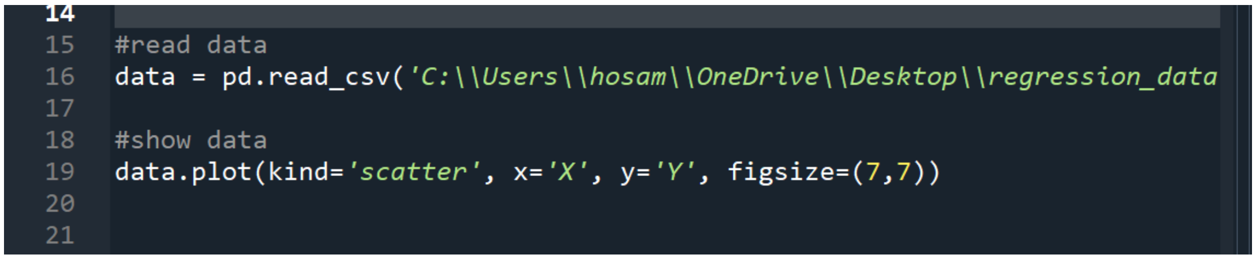
```python
#import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

1) Import Libraries.

Numpy for Matrices creation, operations

Pandas for Read our data

Matplotlib.pylot for Draw graphs



```python

#read data
data = pd.read_csv('C:\\Users\\hosam\\OneDrive\\Desktop\\regression_data

#show data
data.plot(kind='scatter', x='X', y='Y', figsize=(7,7))


```

2) Read the data.

Load data into (DATA) using Pandas function (read_csv).

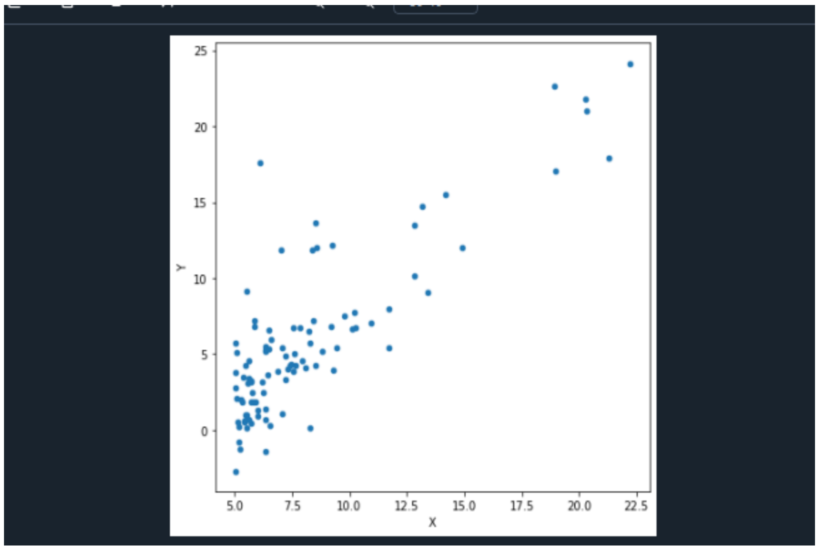
Three arguments:

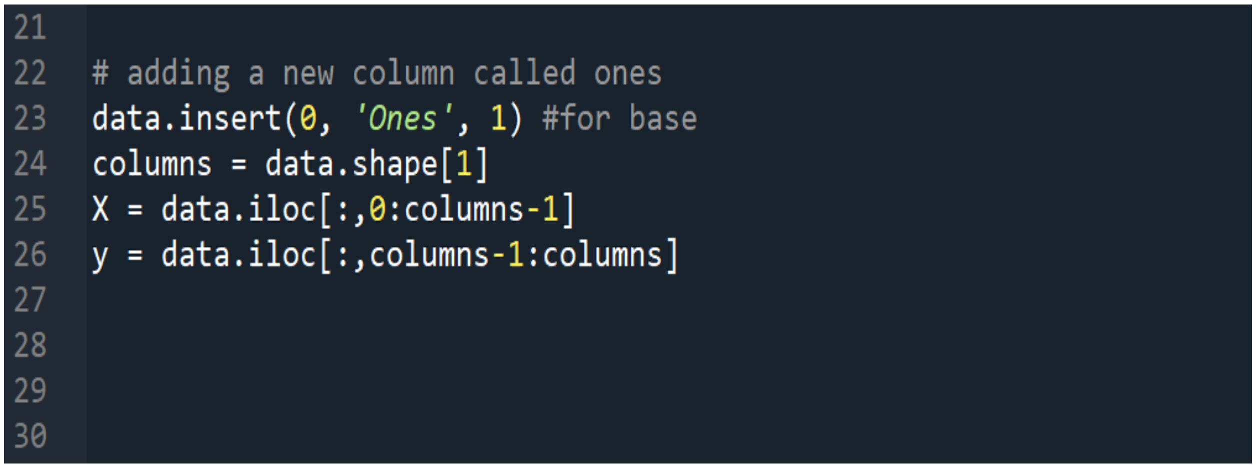
1. The path of file

2. Header is the first row of the file, sometimes first row has the names of Columns, but in our case we don't have it so, header → null

3. Names of columns

Returns a line graph containing data from every row in the DataFrame.



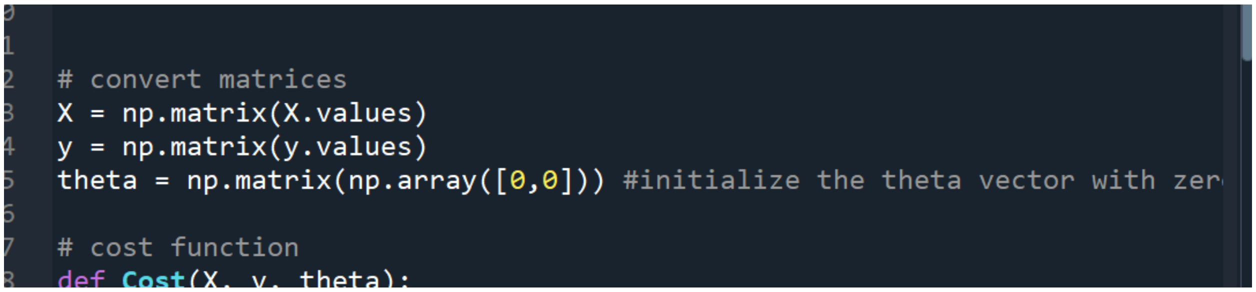


```python

# adding a new column called ones
data.insert(0, 'Ones', 1) #for base
columns = data.shape[1]
X = data.iloc[:,0:columns-1]
y = data.iloc[:,columns-1:columns]




```

3) Make matrix X and y.

First insert column with ones for base in first column, then separate data in X and y.

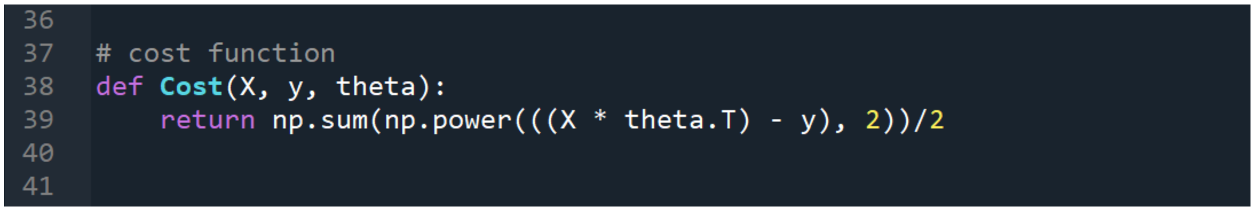
X for data and Y for target.



```python

# convert matrices
X = np.matrix(X.values)
y = np.matrix(y.values)
theta = np.matrix(np.array([0,0])) #initialize the theta vector with zer

# cost function
def Cost(X, y, theta):
```

….. (Count for 3).

Also when we make matrix we create theta matrix with initialize of zeros

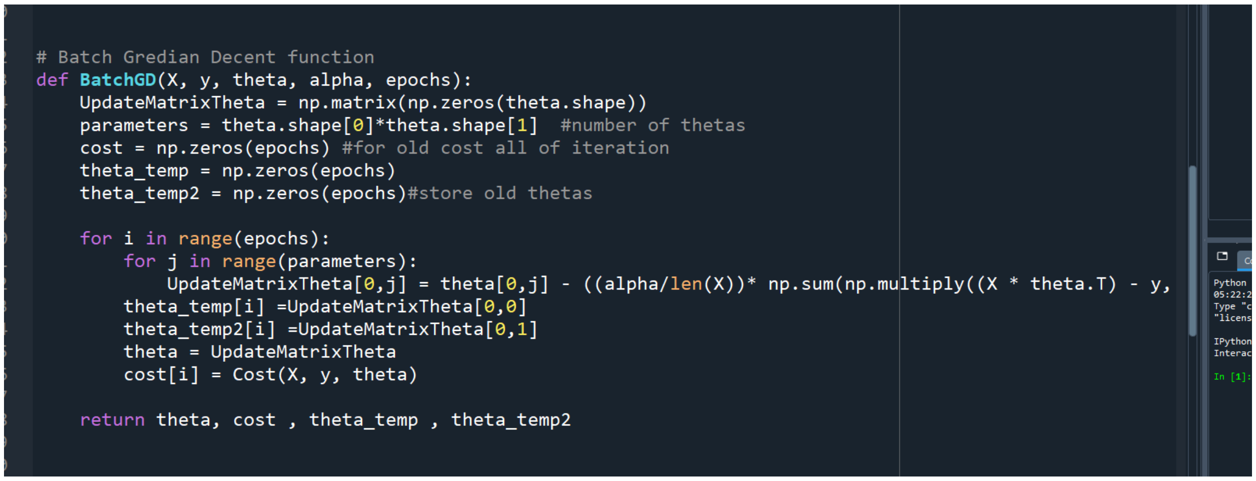


```python

# cost function
def Cost(X, y, theta):
    return np.sum(np.power(((X * theta.T) - y), 2))/2

```

4) Cost function

For knowing the lost and distance of true value, when it is lower when it is better.

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{n}(h_\theta(x^{(i)}) - y^{(i)})^2.$$



```python
# Batch Gredian Decent function
def BatchGD(X, y, theta, alpha, epochs):
    UpdateMatrixTheta = np.matrix(np.zeros(theta.shape))
    parameters = theta.shape[0]*theta.shape[1]  #number of thetas
    cost = np.zeros(epochs) #for old cost all of iteration
    theta_temp = np.zeros(epochs)
    theta_temp2 = np.zeros(epochs)#store old thetas

    for i in range(epochs):
        for j in range(parameters):
            UpdateMatrixTheta[0,j] = theta[0,j] - ((alpha/len(X))* np.sum(np.multiply((X * theta.T) - y,
        theta_temp[i] =UpdateMatrixTheta[0,0]
        theta_temp2[i] =UpdateMatrixTheta[0,1]
        theta = UpdateMatrixTheta
        cost[i] = Cost(X, y, theta)

    return theta, cost , theta_temp , theta_temp2
```

5) Batch gradient

We will create matrix 1*2 for matrix and its values = Zero → UpdateMatrixTheta

Put in PARAMETERs var num. of col. of theta matrix.

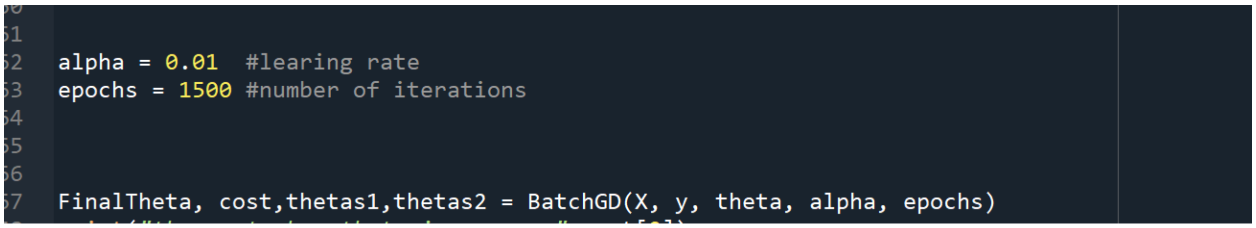
COST var = matrix with 1500 zero to store cost_value_history.

Theta temp for store all values of first element of thetas (one and two).

Second loop will loop for two times j = 0,1 and multiply every (h(x) – y) in X0 values and (h(x) – y) in X1 into → term,

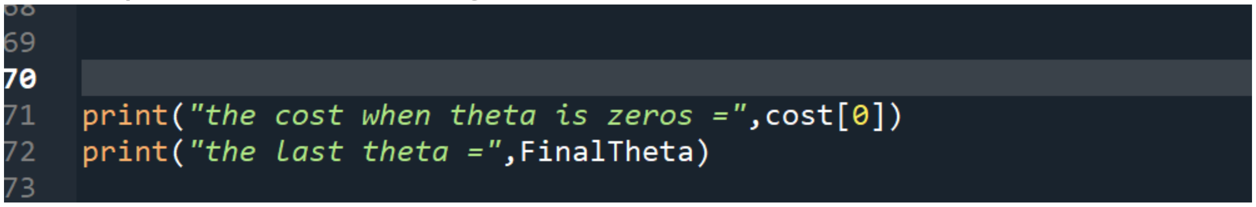
then compute both of new theta values, in first loop after inner loop we update theta(s) and store cost value for these theta(s), Finally we return the Final Theta(s) and cost_value_history and temp thetas one and two .

$$\begin{aligned}
\frac{\partial}{\partial \theta_j} J(\theta) &= \frac{\partial}{\partial \theta_j} \frac{1}{2} \left(h_\theta(x) - y\right)^2 \\
&= 2 \cdot \frac{1}{2} \left(h_\theta(x) - y\right) \cdot \frac{\partial}{\partial \theta_j} \left(h_\theta(x) - y\right) \\
&= \left(h_\theta(x) - y\right) \cdot \frac{\partial}{\partial \theta_j} \left(\sum_{i=0}^{d} \theta_i x_i - y \right) \\
&= \left(h_\theta(x) - y\right) x_j
\end{aligned}$$



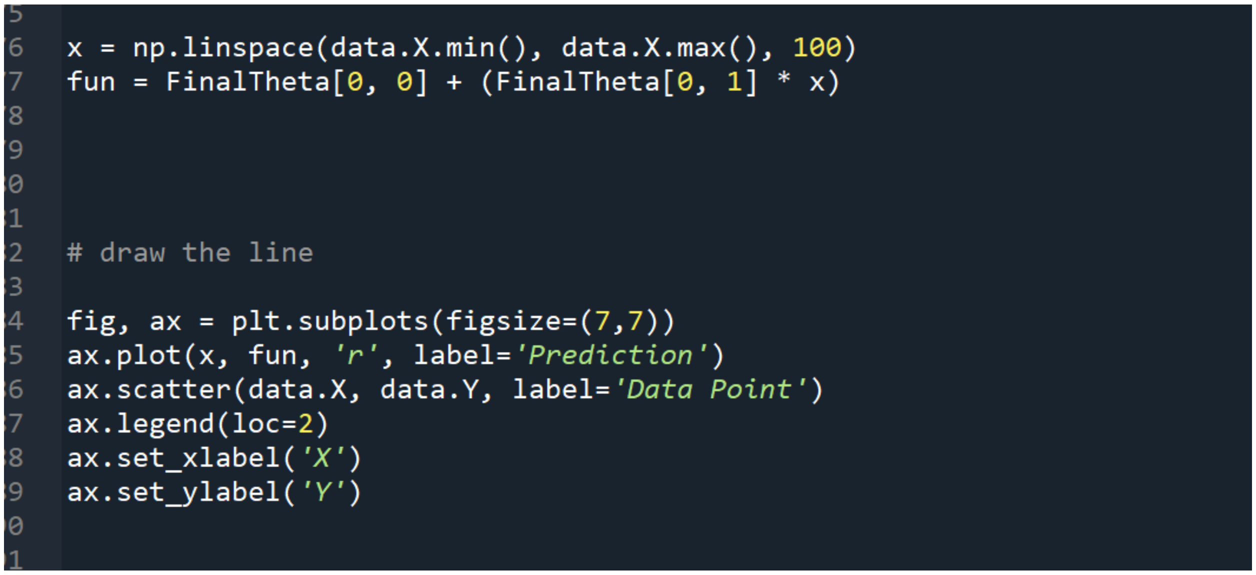
```
alpha = 0.01  #learing rate
epochs = 1500 #number of iterations



FinalTheta, cost,thetas1,thetas2 = BatchGD(X, y, theta, alpha, epochs)
```

6)Initialize Learning Rate and number of Epochs



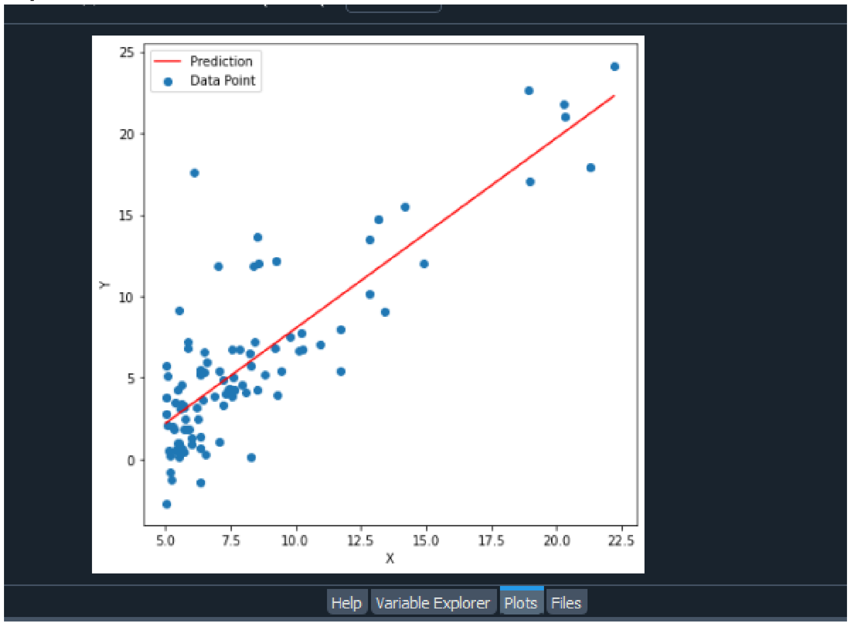
```

print("the cost when theta is zeros =",cost[0])
print("the last theta =",FinalTheta)
```

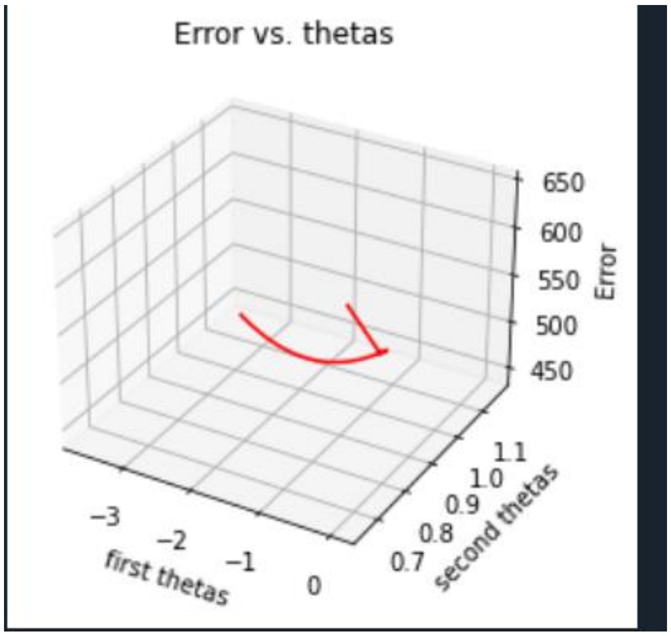
7)print the first cost and last theta.



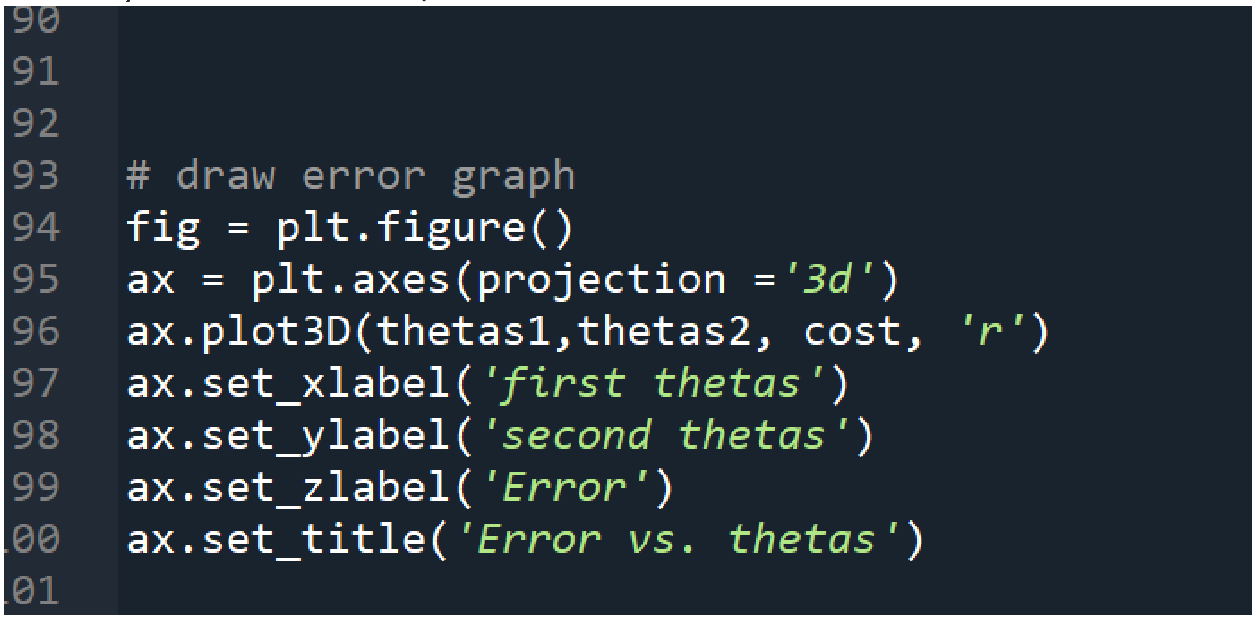
```
x = np.linspace(data.X.min(), data.X.max(), 100)
fun = FinalTheta[0, 0] + (FinalTheta[0, 1] * x)




# draw the line

fig, ax = plt.subplots(figsize=(7,7))
ax.plot(x, fun, 'r', label='Prediction')
ax.scatter(data.X, data.Y, label='Data Point')
ax.legend(loc=2)
ax.set_xlabel('X')
ax.set_ylabel('Y')
```

8) Plot data with the best-fit line



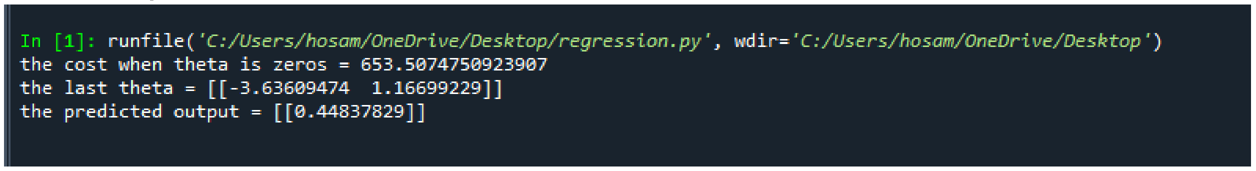


9) Plot Error Graph



```
# draw error graph
fig = plt.figure()
ax = plt.axes(projection ='3d')
ax.plot3D(thetas1,thetas2, cost, 'r')
ax.set_xlabel('first thetas')
ax.set_ylabel('second thetas')
ax.set_zlabel('Error')
ax.set_title('Error vs. thetas')
```

10)show terminal output



```
In [1]: runfile('C:/Users/hosam/OneDrive/Desktop/regression.py', wdir='C:/Users/hosam/OneDrive/Desktop')
the cost when theta is zeros = 653.5074750923907
the last theta = [[-3.63609474  1.16699229]]
the predicted output = [[0.44837829]]
```