Machine learning

Assignment 1

Name: Mohamed Amr El-Bahrawy
ID: 40-3945

Table of contents:

1) Methodology:

This assignment is divided to 4 parts

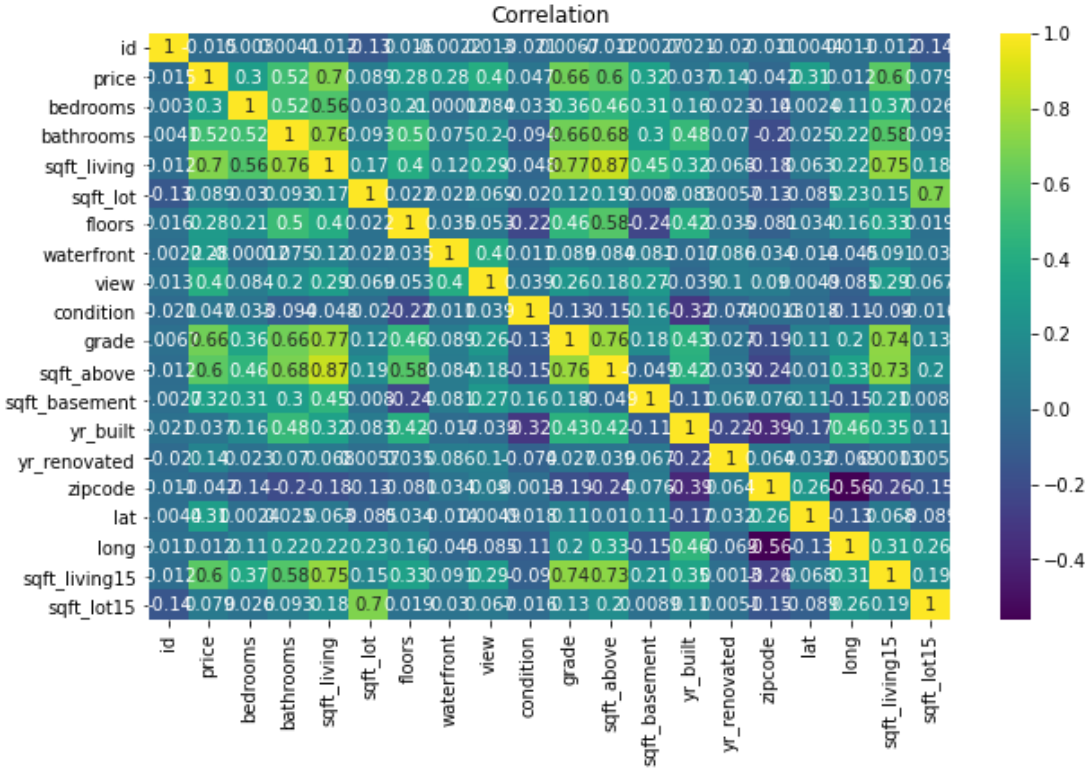
- The first part is reading the data then splitting the data to three sets training set, validation set and testing set.
- Second part is calculating the computation cost of every degree using training set and knowing the thetas which will make this computation cost low as possible.
- Third part is calculating the computation cost using validation set to ensure that hypothesis function and the degree of this function that you calculated from training set is generalized.
- The last part is testing using testing set

Now I will explain every part in details

1.1 splitting data:

First we have an excel sheet we need to read it by using pandas and this function you will insert in it the path of your excel sheet, then after reading the data you will split this data to three sets the first set is training set with percentage 60%, the second set is validation set with percentage 20%, the third set is testing set with percentage 20%.

After that we need to find the correlation between the price and the other features by using correlation function and this figure explain the relation between the price and other features.



We found that 7 features have the most effect on the price which are bedrooms, bathrooms, sqft_living and every feature has probability greater than 0.4.

We have the features and indexes of each feature in the array, also we have each set with its data so now we can convert every set to numpy to deal with it as an array.

1.2 training part:

In this part I divided the training set to two parts train_x that has the data of every feature and train_y has the price column data then we concatenate ones to train_x because as said in the lectures X0 is always equal to ones.

Then we used two functions

1. computeCostMulti(X, y, theta) is responsible for computing the cost of every degree and has inputs train_x, train_y and thetas
   by using this equation:

   ```
   h = np.dot(X, theta)

   J = (1/(2 * m)) * np.sum(np.square(np.dot(X, theta) - y))
   ```

2. GradientDescentMulti(X, y, theta, alpha, num_iters): is responsible for calculating the theta and update it to minimize the computation cost.
   By using this equation:

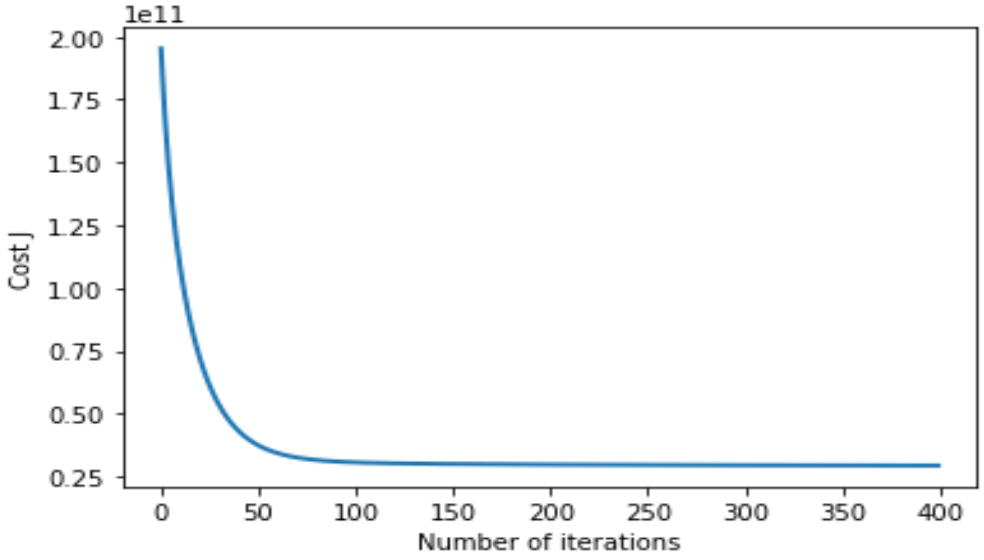
   ```
   theta = theta - (alpha / m) * (np.dot(X, theta) - y).dot(X)

   J_history.append(computeCostMulti(X, y, theta))
   ```

Now we will call gradient Descent function for each degree to know the best thetas that will make our computation cost is low as possible and the computation cost will be an array we will take the last element in the array because it's the smallest one we will loop 400 iterations and it will append every J for 400 iterations in this array.

We will compare all computation values that produced for every degree and the lowest computation value we will use its hypothesis function and it has the best thetas

to minimize the cost.



So after computing the cost the best hypothesis function which has the 8 degree.

We will confirm this by using validation set.

1.3 validation part:

In this part we will divide our validation set to two parts validation_x that contains all features and validation_y that contains the price column data.

We used only one function computeCostMulti(X, y, theta): is responsible for calculating the error or the cost for hypothesis function by using this equations

$$h = np.dot(X, theta)$$

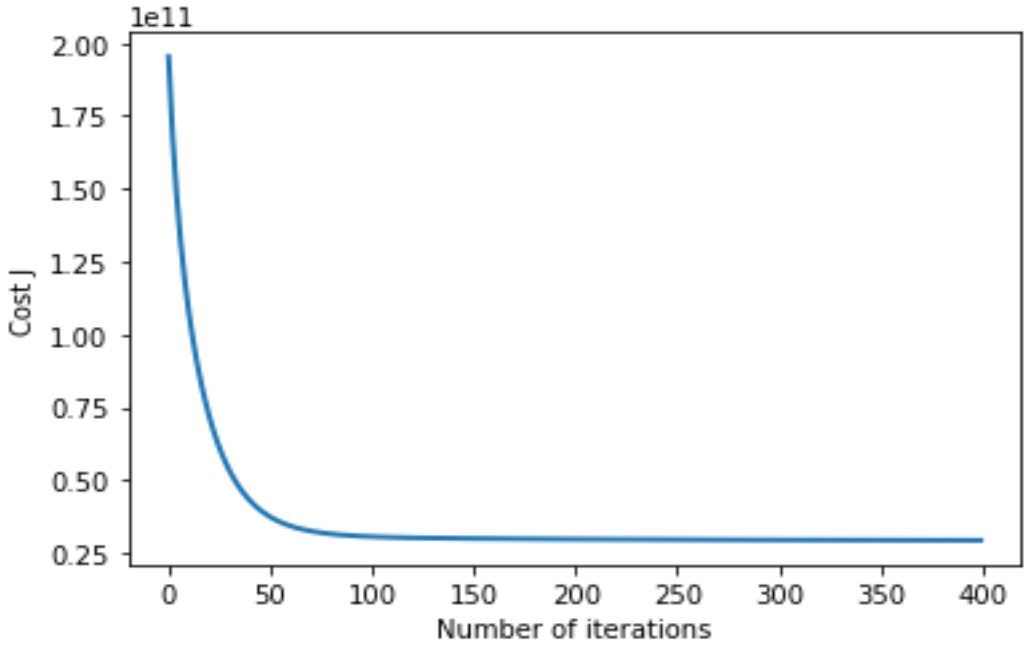
$$J = (1/(2 * m)) * np.sum(np.square(np.dot(X, theta) - y))$$

we used in this function validation_x as, validation_y and thetas for every degree that you calculated in gradient descent and compare the computation cost for every degree as inputs.

If the lowest computation cost is the same degree as training set so this hypothesis is generalized.

And I found that the lowest computation cost is the same degree which is equal to 8.

Now let’s move to testing set.



1.4 testing part:

In this part we will use testing set and we will divide this set to two parts test_x and test_y. test_x has all features and test_y has the price column data.

We used in this part only one function computeCostMulti(X, y, theta): which is responsible for testing your hypothesis function and see how is far from the output y

And it uses this equations:

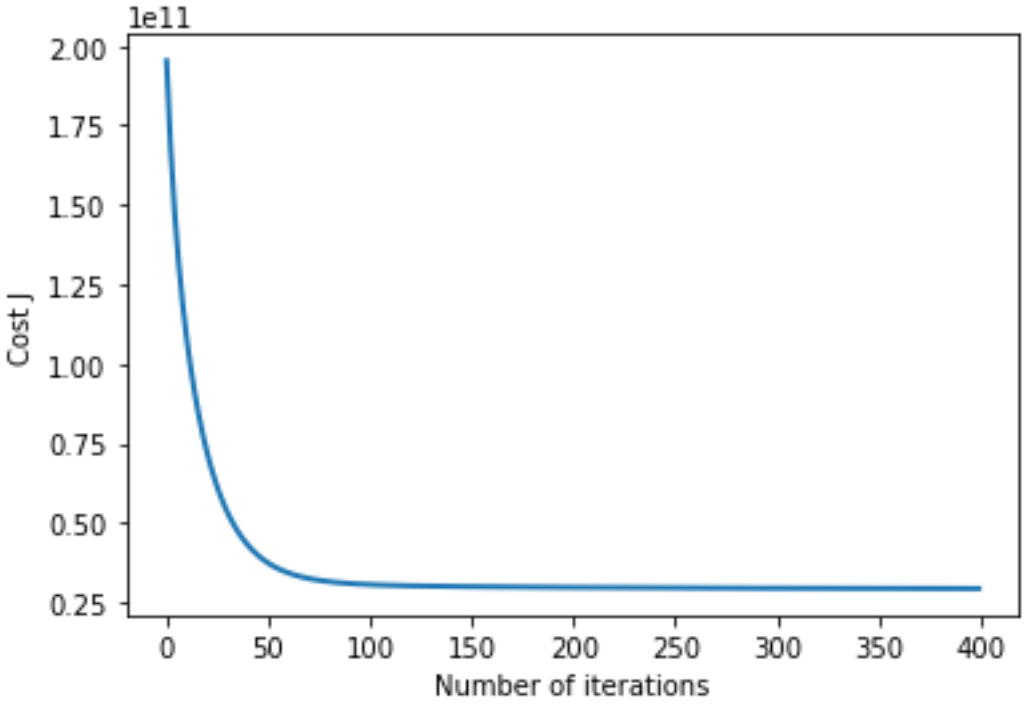
$$h = np.dot(X, theta)$$

$$J = (1/(2 * m)) * np.sum(np.square(np.dot(X, theta) - y))$$

This function takes input test_x, test_y and thetas this thetas that you calculated from gradient descent and and confirm it using validation set, also we will use the hypothesis function which has the same degree that you calculated from training set and confirmed by validation set.

Now we tested the error and this is the computation cost

25596092582.814125


1e11
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
Cost J
0
50
100
150
200
250
300
350
400
Number of iterations