Homework 1

Problem 1 (40 points)

Develop a code that run linear regression with a gradient descent algorithm for each of the explanatory variables in isolation. In this case, you assume that in each iteration, only one explanatory variable (either X1, or X2, or X3) is explaining the output. Basically, you need to do three different training, one per each explanatory variable. For the learning rate, explore different values between 0.1 and 0.01 (your choice). Initialize your parameters to zero (theta to zero).

1. Report the linear model you found for each explanatory variable.

   My learning rate is 0.03 (which is alpha)

   Number of iterations is 1000

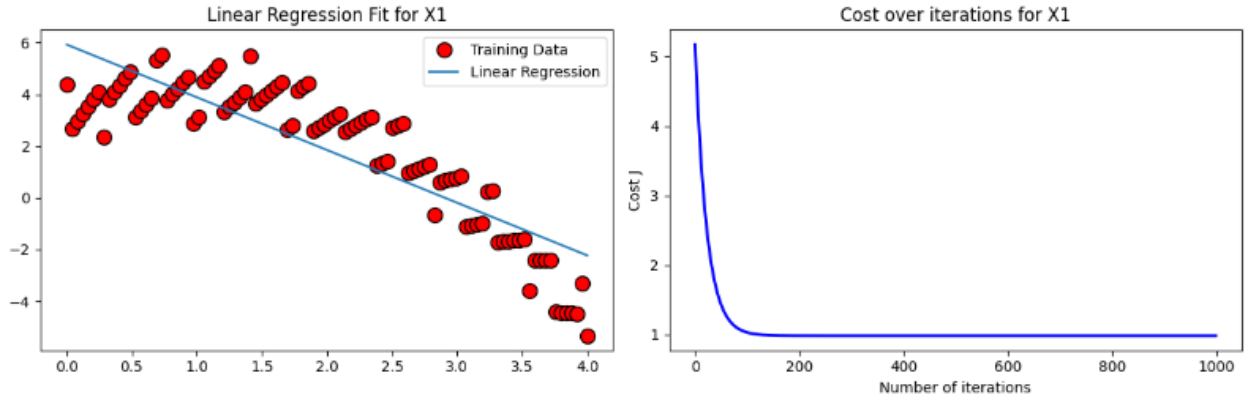
   X1: array([5.92054437, -2.03545475]),

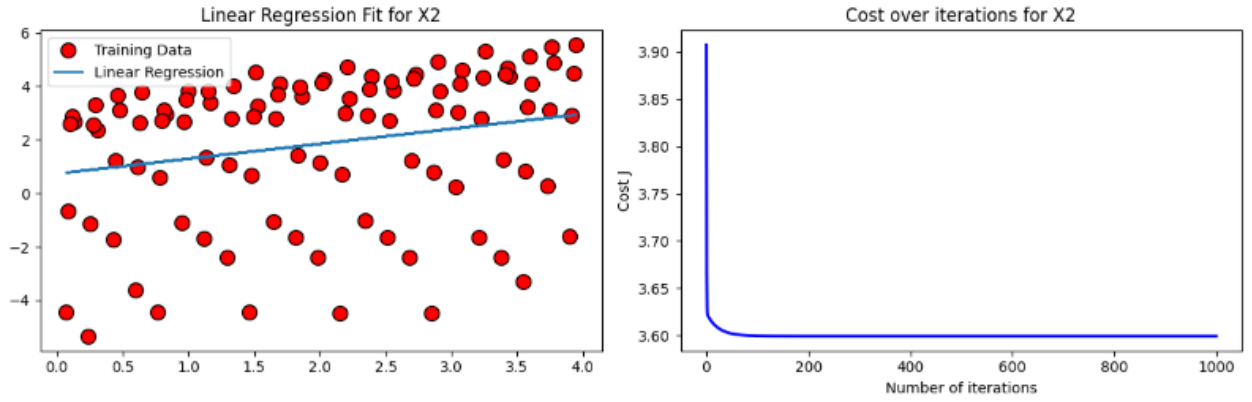
   X2: array([0.73548849, 0.55783022]),

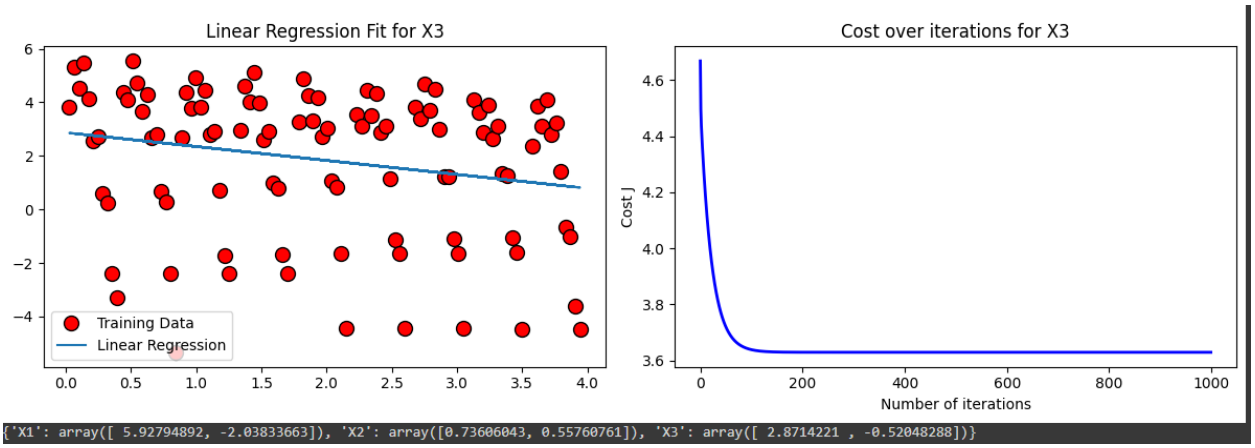
   X3: array([2.86836127, -0.51927234])}

2. Plot the final regression model and loss over the iteration per each explanatory variable.

1 & 2







3. Which explanatory variable has the lower loss (cost) for explaining the output (Y)?

=> You can find the explanatory variable with the lowest loss by comparing the last value in their J_history.

4. Based on your training observations, describe the impact of the different learning rates on the final loss and number of training iterations.

=> adjusting the learning rate which is ‘alpha’ in my code. Can lead to the following:
1) the higher the value is the convergence can be faster which leads to more scopic approach (more accurate scope)

2) the lower the value is the more iterations it needs to converge.

Problem 2 (60 points)

This time, run linear regression with gradient descent algorithm using all three explanatory variables. For the learning rate, explore different values between 0.1 and 0.01 (your choice). Initialize your parameters (theta to zero).

1. Report the final linear model you found the best.
2. Plot loss over the iteration.

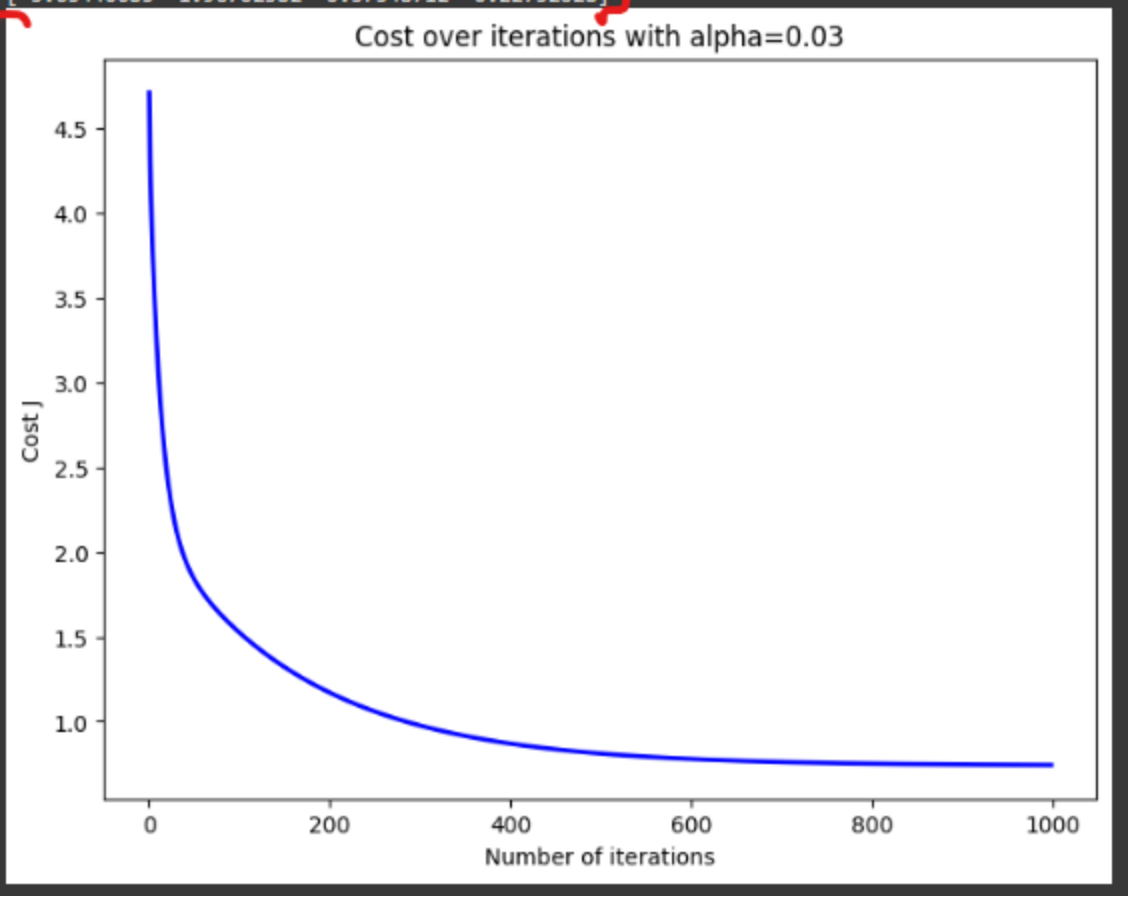
1 and 2

```
X = data.iloc[:, 0:3].values
X = np.column_stack((np.ones(len(X)), X))  # add a bias term
y = data.iloc[:, 3].values
theta = np.zeros(4)
alpha = 0.03  # Chosen value between 0.1 and 0.01, you can change this to experiment
num_iters = 1000

theta_final, J_history = gradient_descent(X, y, theta, alpha, num_iters)
print(theta_final)
plt.figure(figsize=(8, 6))
plt.plot(range(num_iters), J_history, '-b', linewidth=2)
plt.xlabel('Number of iterations')
plt.ylabel('Cost J')
plt.title(f'Cost over iterations with alpha={alpha}')
plt.show()
```

Model

```
[ 5.05440685 -1.96702382  0.57548712 -0.22752628]
```



3. Based on your training observations, describe the impact of the different learning rates on the final loss and number of training iteration.

=> higher learning rates may converge faster but risk overshooting, while lower rates might be more steady but require more iterations. The chosen learning rate of 0.03 is a balanced choice.

4. Predict the value of y for new (X1, X2, X3) values (1, 1, 1), for (2, 0, 4), and for (3, 2, 1)

```
[8]  def predict(X, theta):
         return np.dot(X, theta)

     # Values
     data_points = np.array([
         [1, 1, 1],
         [2, 0, 4],
         [3, 2, 1]
     ])

     # Add bias term
     data_points = np.column_stack((np.ones(len(data_points)), data_points))

     predictions = predict(data_points, theta_final)
     print(predictions)
```

```
[3.43534387 0.21025408 0.07678334]
```

Links: source code and
https://github.com/GGYBlank/MLHWCore/tree/main/Homework/Homework%201