ECGR 4105 Homework 0

Nahush D. Tambe – 801060297

https://github.com/Ntambe25

Problem # 1

1. The Linear models for each of the explanatory variable X1, X2, and X3 are as follows:

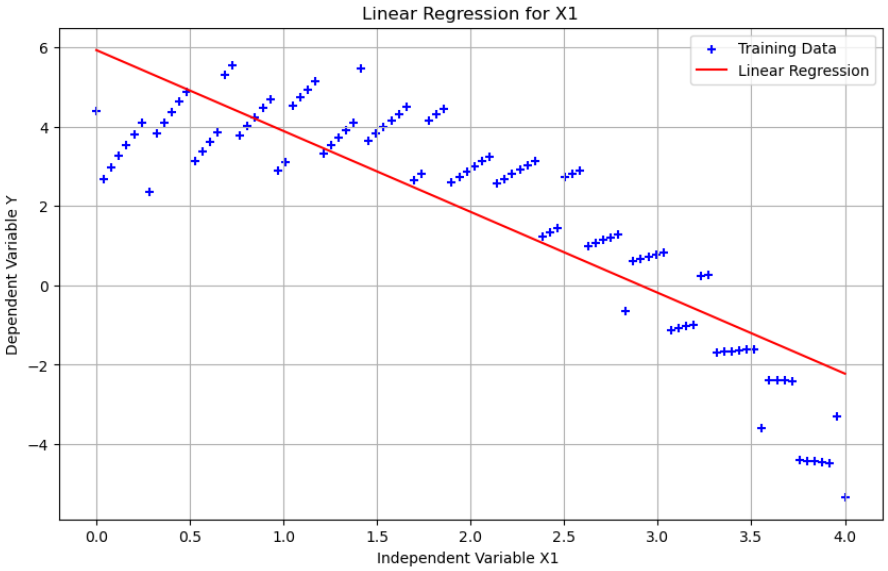

Figure 1: Linear Regression with X1

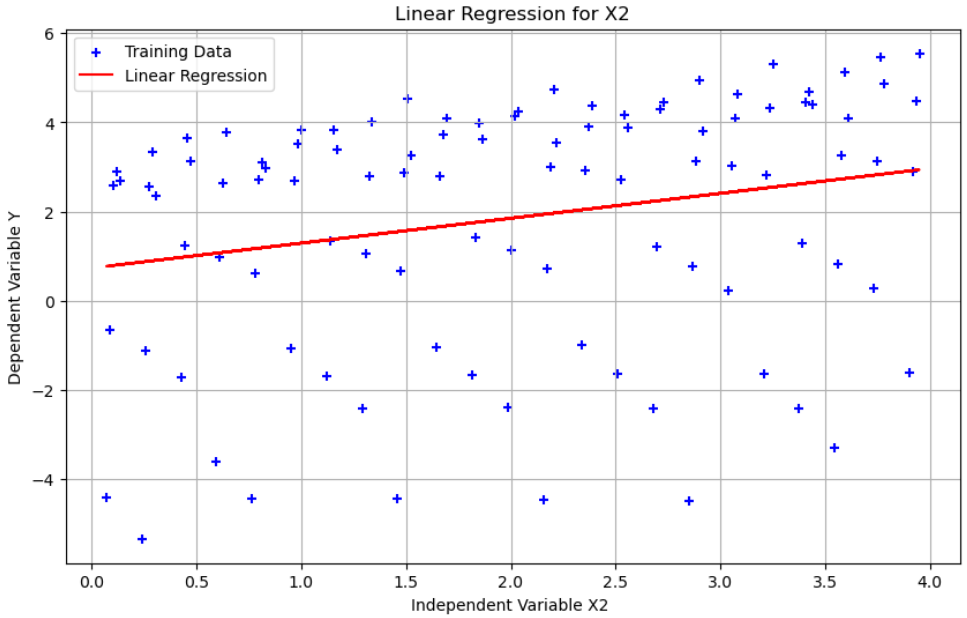

Figure 2: Linear Regression with X2

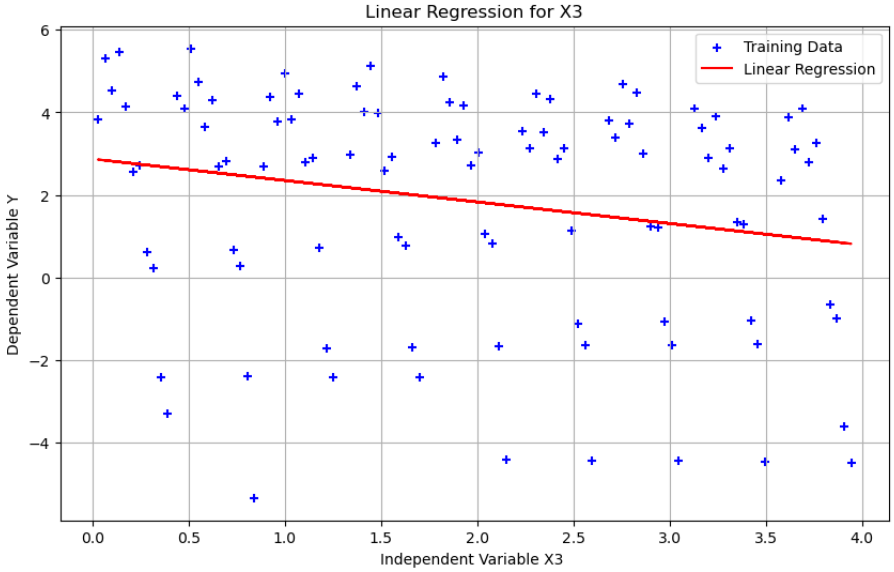

Figure 3: Linear Regression with X3

For creating the following 3 figures, first the X variable was loaded, then changed to a 2D array using the reshape function. Next, X0 (a matrix with a single column of 1s) and reshaped X array are stacked horizontally. Using the defined loss and gradient descent function, Linear Regression plot were made.

2. Final Regression Models for loss over the # of iterations for each of the 3 explanatory variables are as follows:

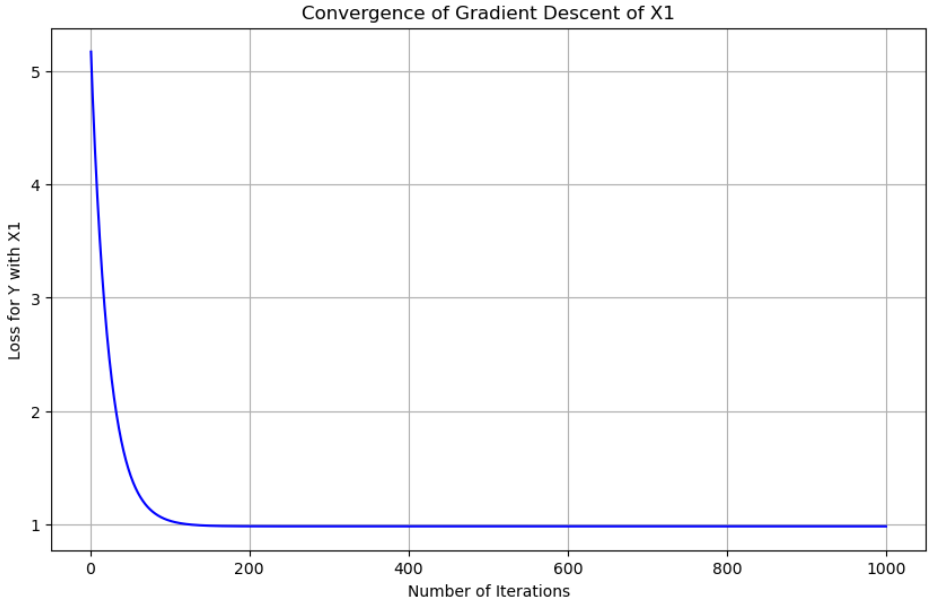

Figure 4: Final Regression with X1

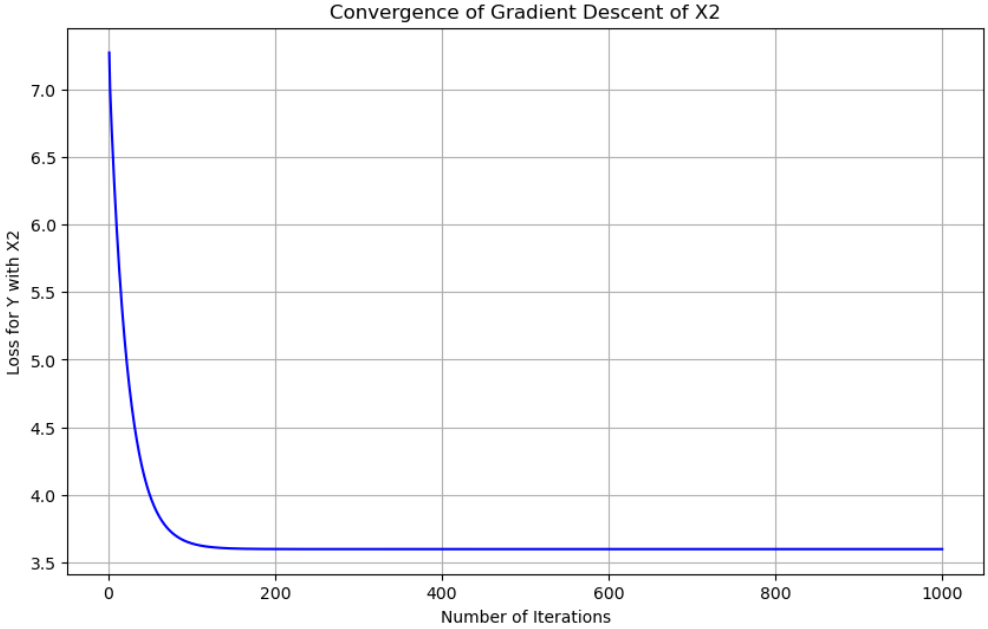

Figure 5: Final Regression with X2

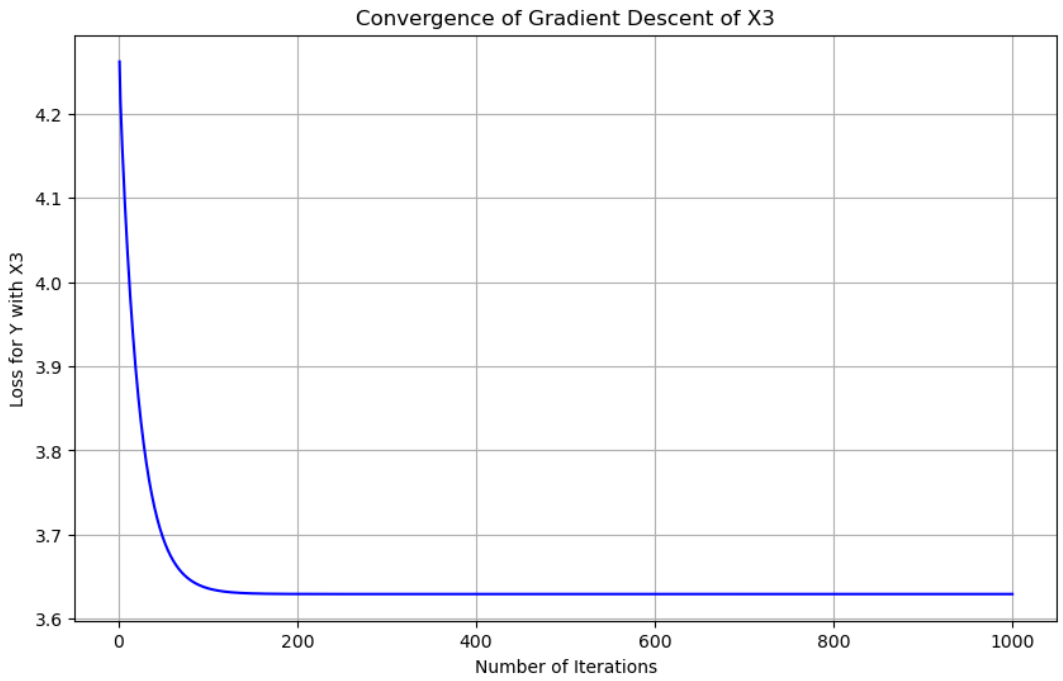

Figure 6: Final Regression with X3

3. After calculating the loss for the output (Y) for each of the 3 explanatory variables, variable X3 has the lowest loss of 4.41. The loss for X1 was 5.52 and loss for X2 was 8.18.
   The method used to calculate loss was using the defined function 'compute_loss' and the statement used was:

```
# Computation of the loss for theta values
loss = compute_loss(X3,Y,theta)


print("The loss for the given values of theta_0 and theta_1 = ", loss)
```

```
The loss for the given values of theta_0 and theta_1 = 4.407877430792258
```

4. After completing first 3 parts of problem 1, the python code was experimented with different alpha (learning rate) values and # of iterations.

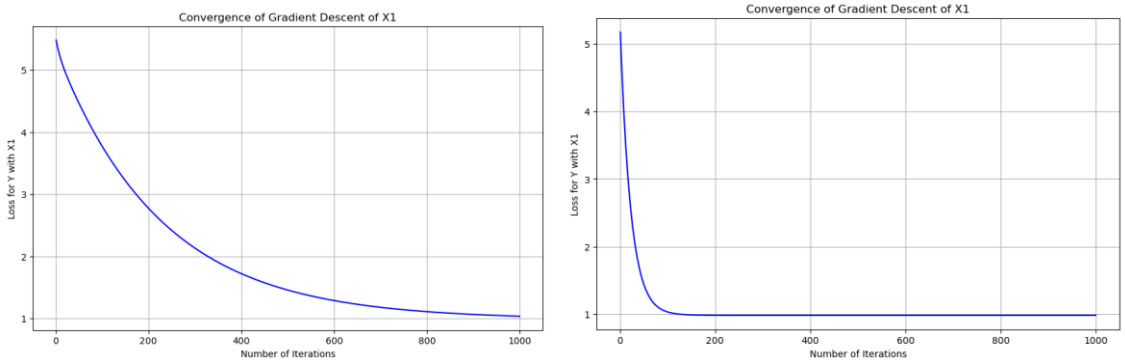

Figure: alpha = 0.01 (LEFT) and alpha = 0.1 (RIGHT)

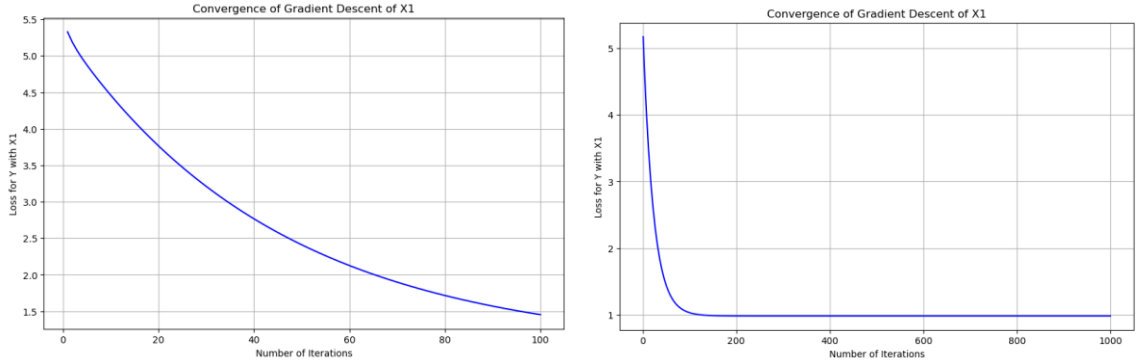

Figure: Iter. = 100 (LEFT) and Iter. = 1000 (RIGHT)

The two figures on the top show the effect of changing learning rate on the Gradient Descent curve, while the two figures at the bottom show the effect of changing the # of iterations on the Gradient Descent curve. Decreasing the learning rate or the # of iterations makes the curve flatten out less slowly, i.e it saturates late, which is not desirable.

Problem # 2

1. The final Linear Model is given below: The process for obtaining the final linear model was same for 3 variables as it was for a single variable. The only difference was instead of stacking 2 arrays, this time 4 arrays were stacked horizontally, and 4 values were used for theta. With the changes, the new defined functions for loss and gradient descent were used.

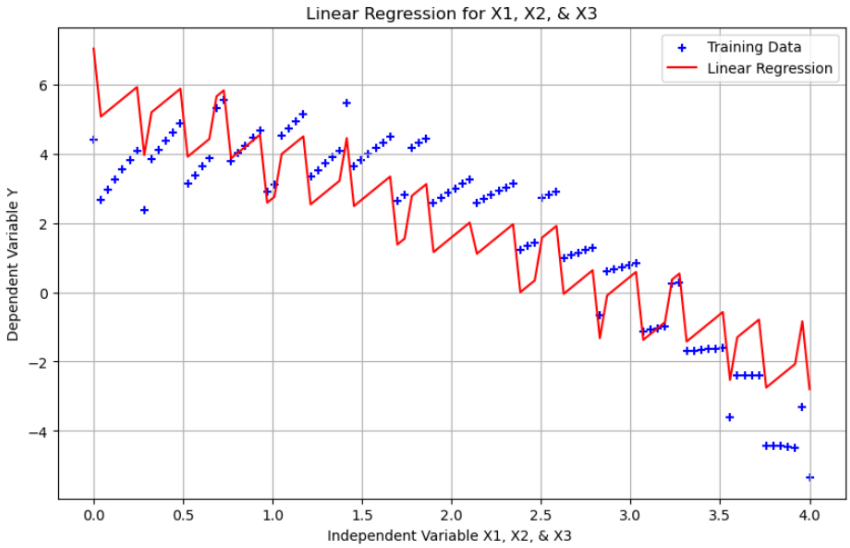

Figure 7: Final Linear Model

2. The plot of loss over the number of iterations is also given below. It was created using the same procedure as for the linear model, but only the plotting parameters included loss_history.

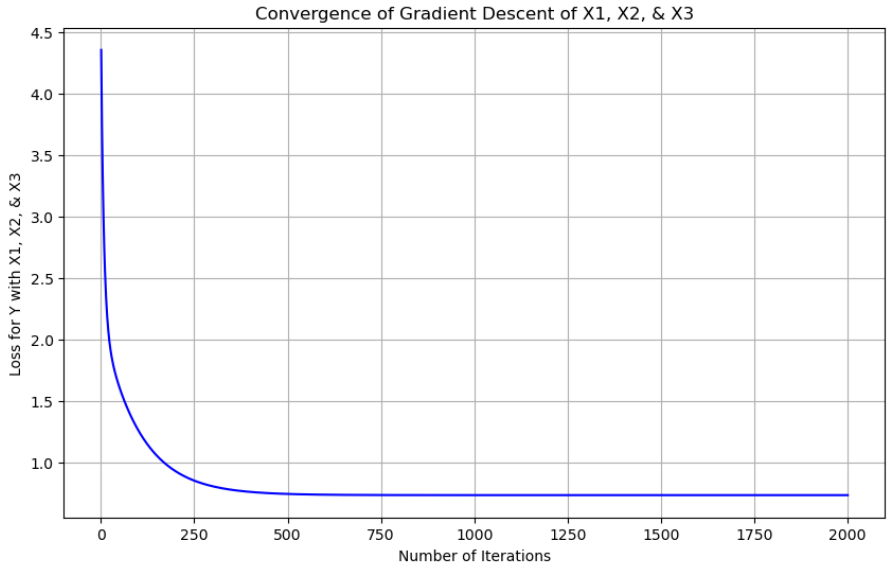

Figure 8: Final Regression Model

3. After obtaining the 2 plots above, the python code was experimented with different alpha (learning rate) values and # of iterations.

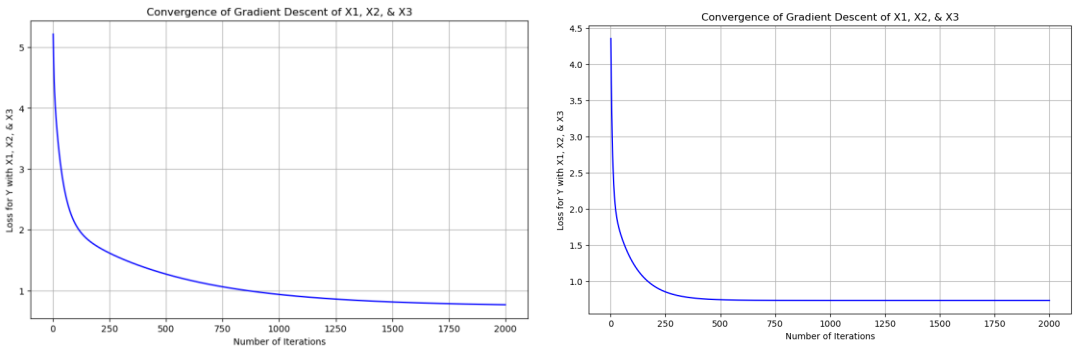

Figure: alpha = 0.01 (LEFT) and alpha = 0.05 (RIGHT)

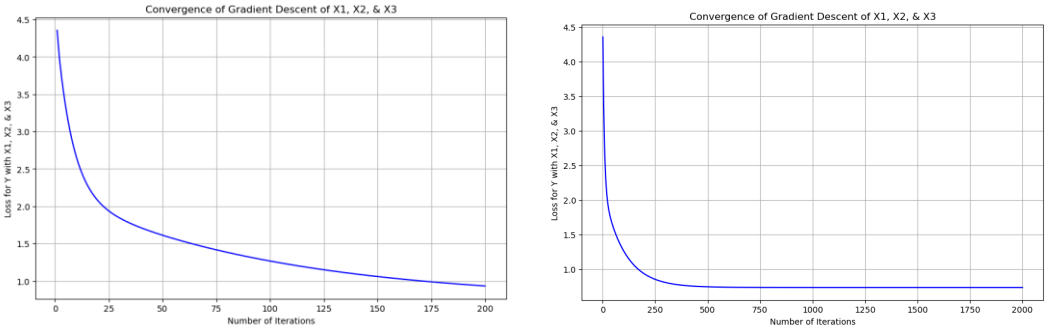

Figure: Iter. = 200 (LEFT) and Iter. = 2000 (RIGHT)

The two figures on the top show the effect of changing learning rate on the Gradient Descent curve, while the two figures at the bottom show the effect of changing the # of iterations on the Gradient Descent curve. Decreasing the learning rate or the # of iterations makes the curve flatten out less slowly, i.e it saturates late, which is not desirable.

4. The predicted values for the output Y for new (X1, X2, X3) values (1, 1, 1), for (2, 0, 4), and for (3, 2, 1) are as follows:

```
In [141]: prediction = theta[0] + 1*theta[1] + 1*theta[2] + 1*theta[3]
          prediction

Out[141]: 3.5772796020480597

In [142]: prediction = theta[0] + 2*theta[1] + 0*theta[2] + 4*theta[3]
          prediction

Out[142]: 0.24429005364325773

In [143]: prediction = theta[0] + 3*theta[1] + 2*theta[2] + 1*theta[3]
          prediction

Out[143]: 0.10251065033649731
```
