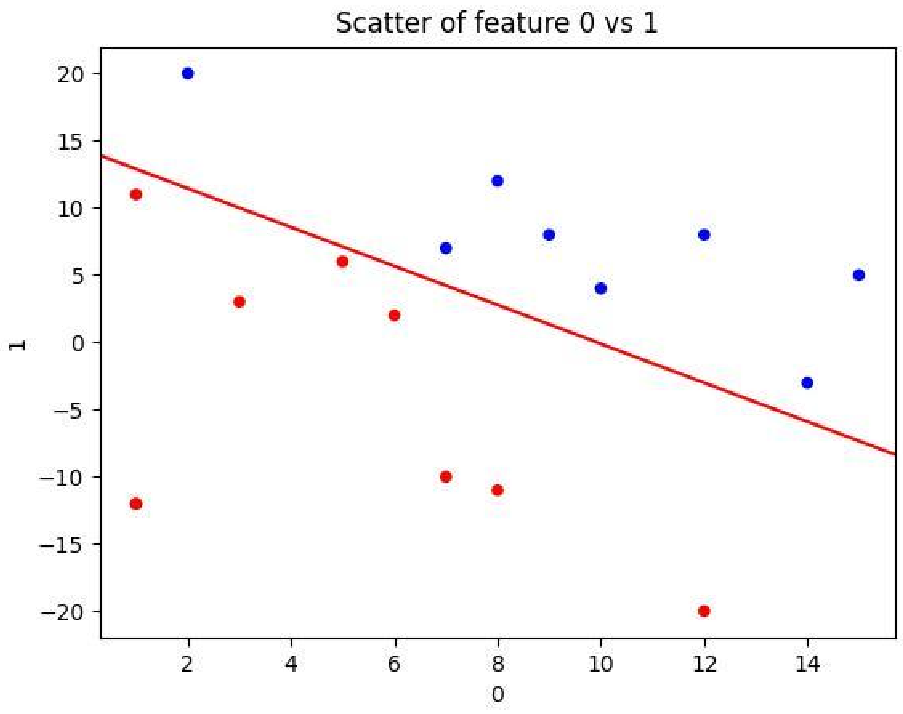
Ahuva Bechhofer
Amb2572

1: Perceptron

The implementation of the perceptron learning algorithm on a given data set with two features. We start with the weights β_0, β_1, β_2 = 0,0,0. And we adjust the weights by iterating through the data and updating the weights when a wrong prediction is made until we reach convergence. The weights that the algorithm found after every full iteration through the data were stored into a new csv file. The resulting decision line based on the weights found for the dataset is shown below.



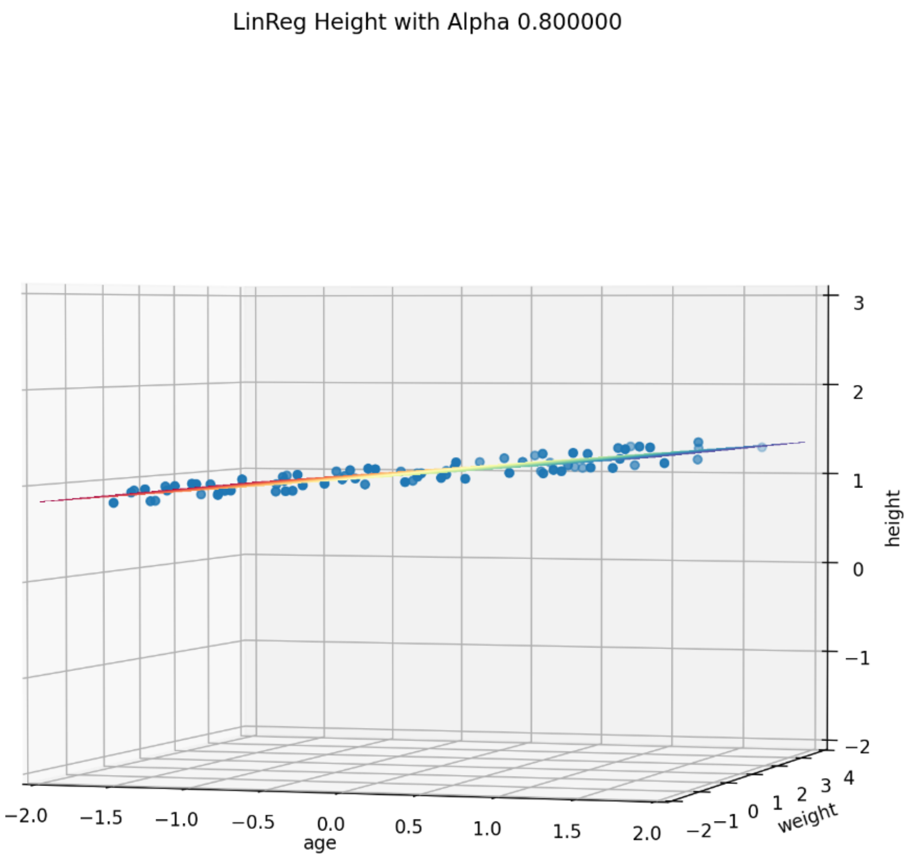
As it can be seen, the perceptron was successful at finding a clear line to separate the data.

2: Linear Regression

The implementation of gradient descent to find a linear regression model to predict the height of an individual given the age and the weight. We first normalized the data so all points are on the same scale. Then we ran gradient descent optimization subcutaneously updating all the weights ($\beta_i = \beta_i \ - \ \alpha/n \ * \ \sum(\beta_0 + \beta_1 x_1 + \beta_2 x_2 - Y_i) \ * x_i$) until we reached convergence.

As for the choice of alpha and the number of iterations, I used a lot of trial and error. First I utilized the visualization tools on hand. Given the code to visualize the 3D graph I was able to tell that alphas from .05 to 1 had the best prediction plane. I then included in my code the empirical risk/MSE for each alpha and I found that for those same alpha ranges the MSE = 0.002461 / 0.00236 which is very close to each other. I then noticed by testing smaller and smaller intervals that the error decreased, approaching 0.8 and then it started to increase again. Therefore I chose $\alpha=0.8$. In testing for the number of iterations, I found that below 100 iterations the error was getting worse but increasing the number of iteration even to 900 did not make a significant difference (there were some small changes in the 17th decimal place but I thought adding another 800 iterations to very slightly improve the error is not worth the computation power of the additional loop iterations.)

Below is the graph of $\alpha=0.8$ and 100 iterations



We can see how this is a nearly perfect prediction with most points falling on the plane or very closely. The error for this is 0.0023640503440440328.