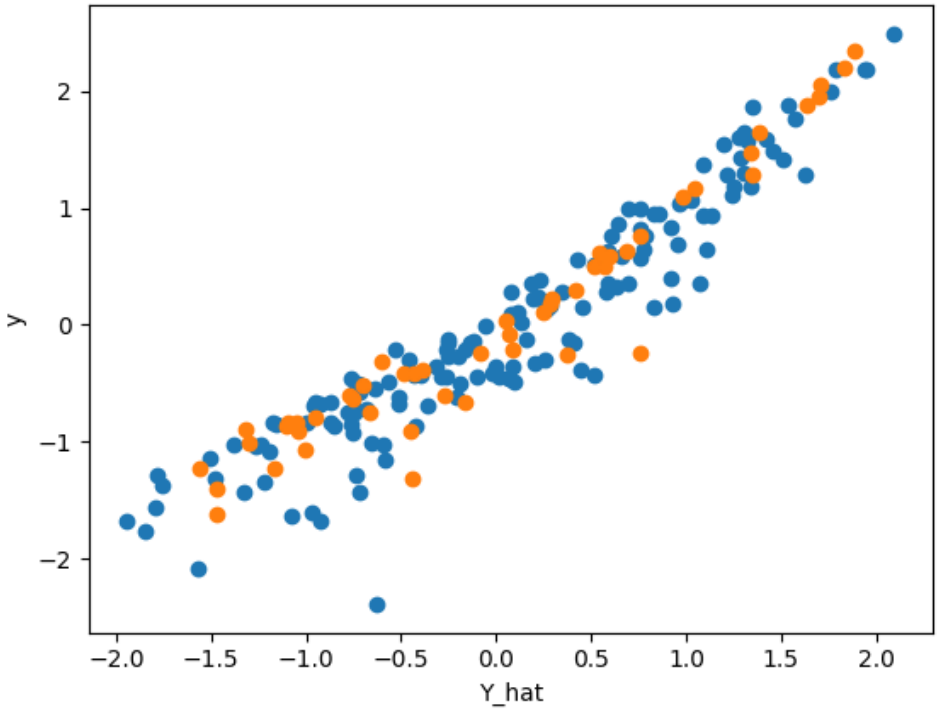
John Gabriel Cabatu-an
CMSC197-1
HW3

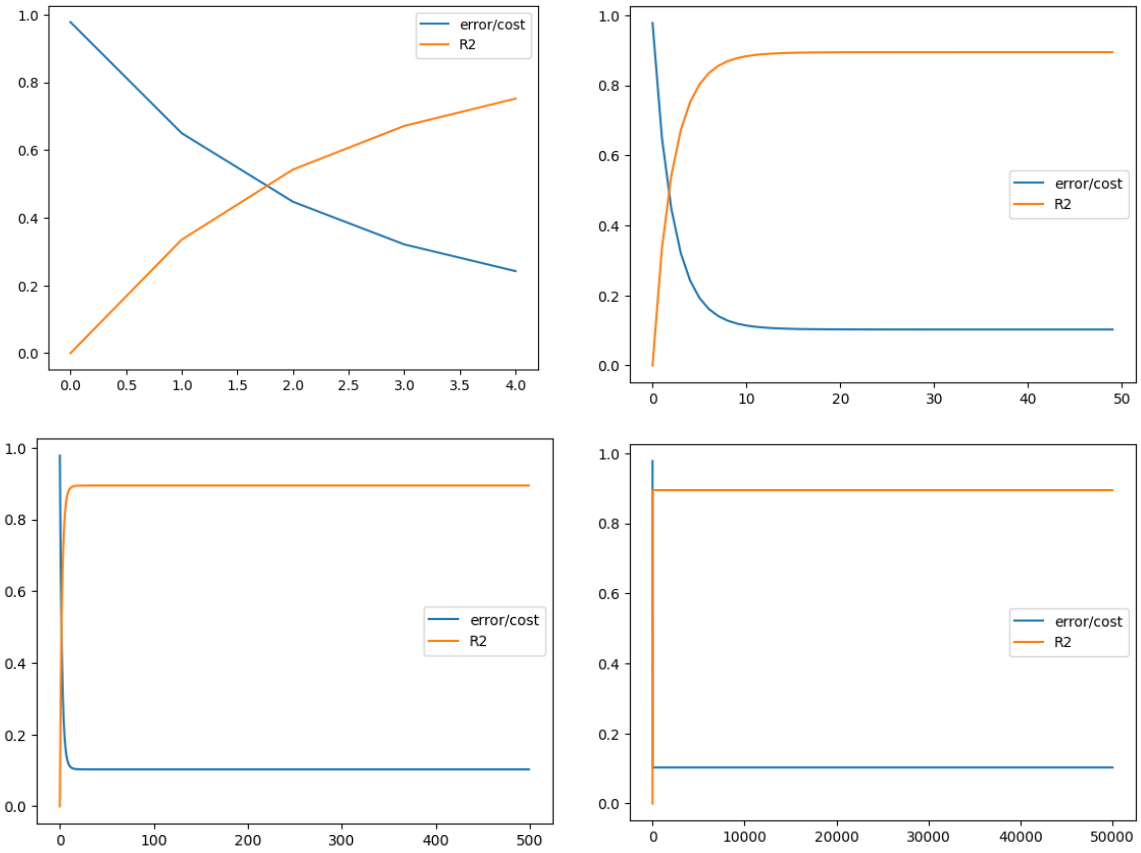
1. What are the optimal weights found by your implemented gradient descent?
    - h_theta(x) = 0.055804713731303364 + 0.7751376567242334 TV + 0.5193436878775318 RADIO + -0.06982243429978127 Newspaper

2. Provide a scatter plot of the 1y (,)5 2 and y (() for both the train and test set. Is there trend? Provide an r2 score (also available in sklearn).



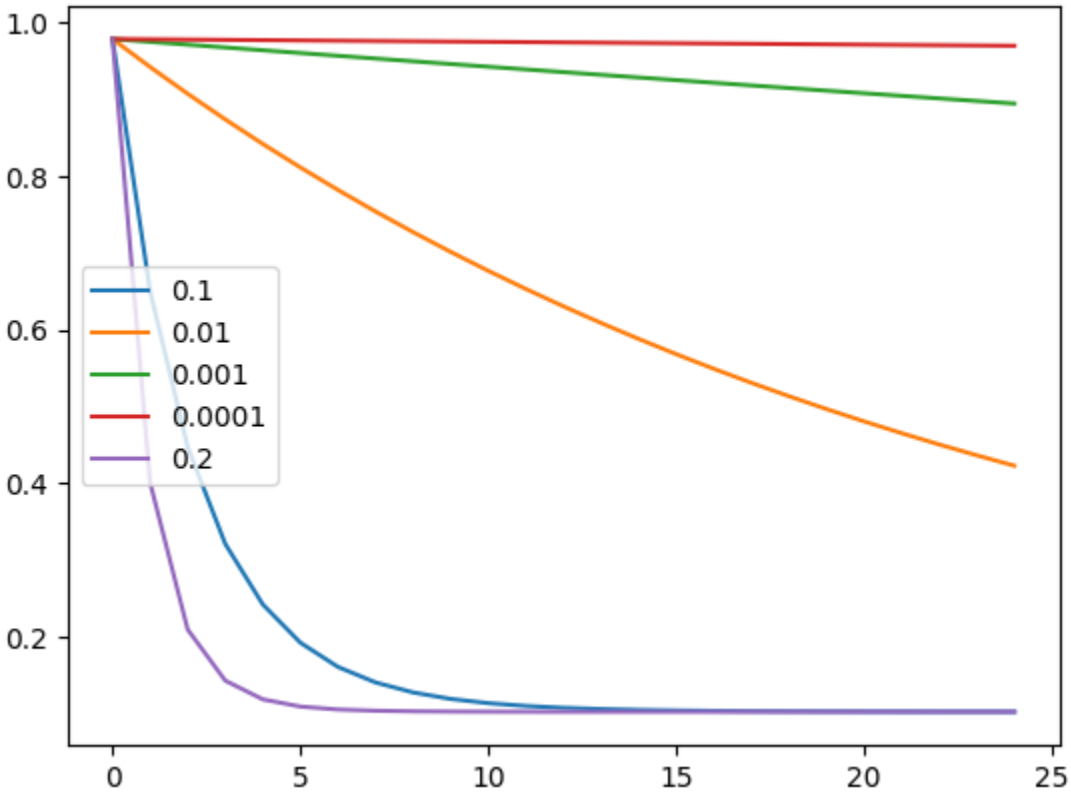
    - train set r2 score: 0.8778908638534562
    - test set r2 score: 0.9174696340278016
    - The scatterplot is observed to have a non-random distribution.

3. What happens to the error, r2, and cost as the number of iterations increase? Show you data and proof. You can alternatively plot your result data for visualization and check until 50000 iterations or more (actually)



- Based on the graphs we can tell that as the iterations increase the error or cost approaches 0 and the R2 score approaches 1 however the graph plateaus at about iteration 20-30

4. Once you determine the optimal number of iterations, check the effect on the cost and error as you change the learning rate. The common learning rates in machine learning include 0.1, 0.01, 0.001, 0.0001, 0.2 but you have the option to include others. Visualize the cost function (vs the optimal number of iterations) of each learning rate in ONLY ONE PLOT. Provide your analysis.



   - As the learning rate decreases the error goes down slowly also. If the learning rate is too small there could be a need for more iterations but if it is too big it might go over the optimal weights.

5. Is there a relationship between the learning rate and the number of iterations?
   - The learning rate depends on the number of iterations to come up with the optimal values for the weights. If there are a lot of iterations but the learning rate is too small it might not come up with the optimal values at all.

6. Compare the results with the results of the ordinary least squares function.
   - using least squares: [0, 0.75303162, 0.53834477, 0.0271173]
   - using gradient_descent: [0.05580471, 0.77513766, 0.51934369, -0.06982243]