Parker Hague

09/08/21

Assignment 1

*See assignment1.ipynb – code block: 1 for this problem’s Python code*

**Problem 1:**

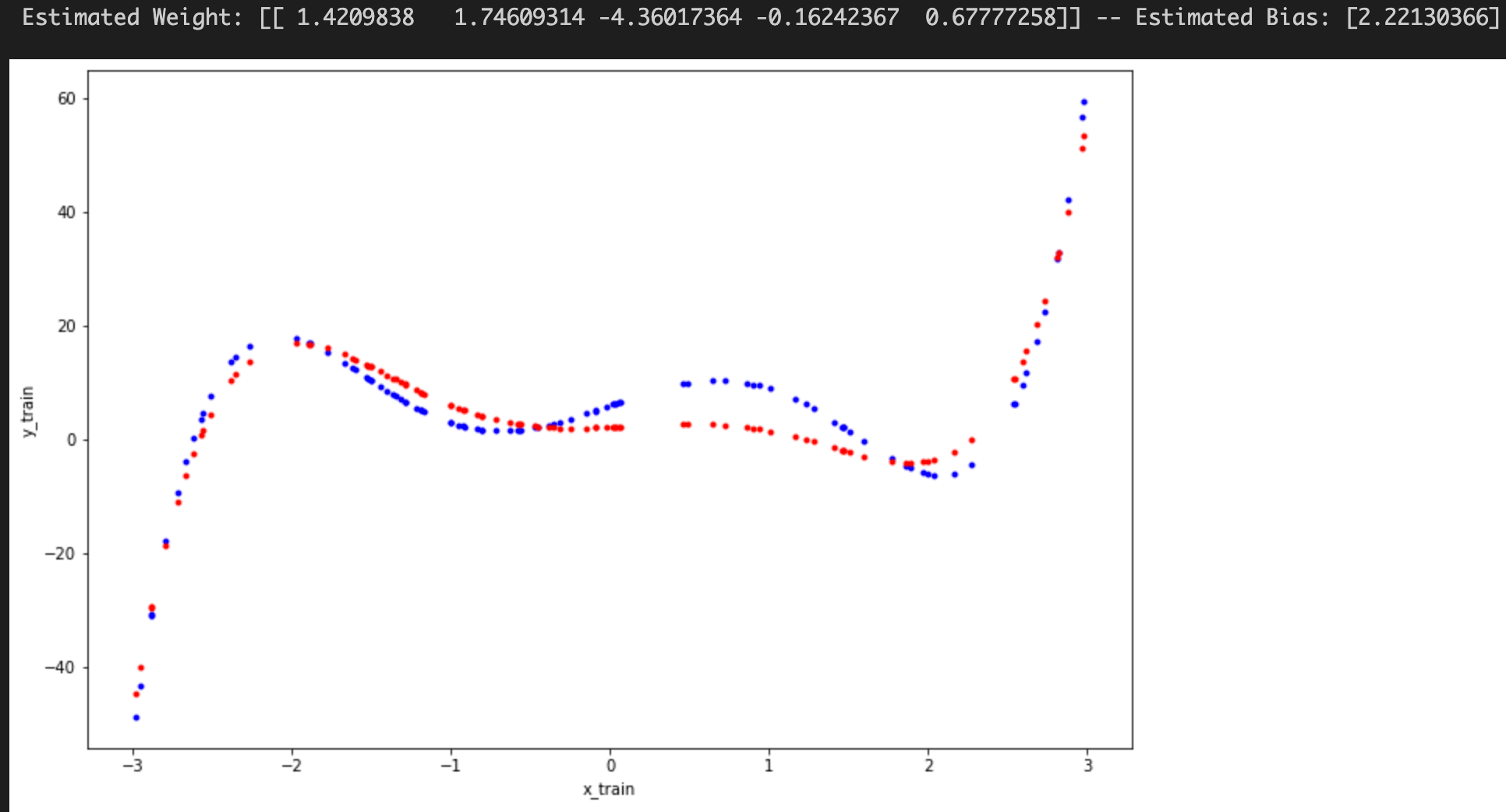
This problem was a little difficult for me. I began by using matplotlib to plot the x\_train and y\_train data. The result was a polynomial function representing the data. I then proceeded to answer the questions asked on problem 1.

1. What clues did the plot give me? After plotting the data, I was immediately able to see a polynomial relationship with the data opposed to a linear one. This immediately suggests to me that I’m going to need to use basis functions on the x\_train data.
2. 1. Is the relationship linear? As I stated previously, no the relationship is definitely not linear. A linear relationship would be a straight line opposed to having (a) curve(s).
   2. Do I need to feature engineering to add non-linearity? Yes. If we want the line to have any sort of non-linearity and actually fit the data, we will have to apply one or many basis functions.
      1. We can use basis functions to engineer these features (as stated multiple times above).
      2. Some easy ones we can plot are the different orders of the x\_train data. For example, we can start with adding an x^2 basis function. Now, look at the graph and see if the model got better or worse. Now, try a third order basis function of x^3. Look at the graph again. Continue this process until you have heuristically found the right amount of basis functions (you may also be wanting to watch the value of your cost function while doing this process).
         1. I plotted the x\_train data starting from the first order to the eleventh. The model got gradually better until we reached around the sixth or seventh order. Then the fit started to become less accurate and the cost started to increase.

Once I was able to create a basic linear regression model, I began adding features via basis functions to introduce non-linearity to the model. I tried all the basis functions from the first order to the eleventh. As stated above, my fit got gradually better until after the fifth order. Then, the cost function began to increase and my model began to worsen.

After finishing my training, I was left with a relationship or function that shows the relationship between x and y. That function is:

Here is a graph of the model plotted with the training data. The blue is the training data and the red is the model I created.



**Final Thoughts:**

Though, I had an error of 387.21 (least mean squares) this function is what modeled the training data most closely. I plotted the test data against the model and was able to confirm that the fit is just as close to the test data as it is to the training data. Though I don’t have a lot of past examples to compare to, my consensus is that this is a fairly good fit. For the sake of this homework, I would consider it a success.

*See assignment1.ipynb – code block: 2 for this problem’s Python code*

**Problem 2:**

1. I was able to calculate the cost for this model using the least mean squares function. The error that I got for this model is 971.98.
2. After modeling the function and doing some calculations, I have determined that garages have the biggest impact on house prices. I determined this through a few ways. First, I took each weight and raised it to the power of the respective order to see which coefficient actually carried the most weight. The next way I determined it was through deduction. Garages range from 0 – 2. This is a very small range compared to the other observations. So a change is this range will carry a lot more weight because there’s less values in the range. For example, a house going from 1 to 2 houses is a 100% increase. Conversely, something like bathrooms has a bigger range and therefore the changes have less impact on the overall model. You cannot successfully only use this feature to predict the price. The outcome would be severely different if one tried.
3. The age of the home has the least effect on the model. I know this because it has the smallest weight of -.07 and the value you get when you raise the weight to the power of the order is practically 0. After removing the feature and retraining the model, I didn’t notice a difference in speed and the model looked almost the complete same. Although, it may be better to keep the observation for the sake of having more data and variety in the data.