Homework 5

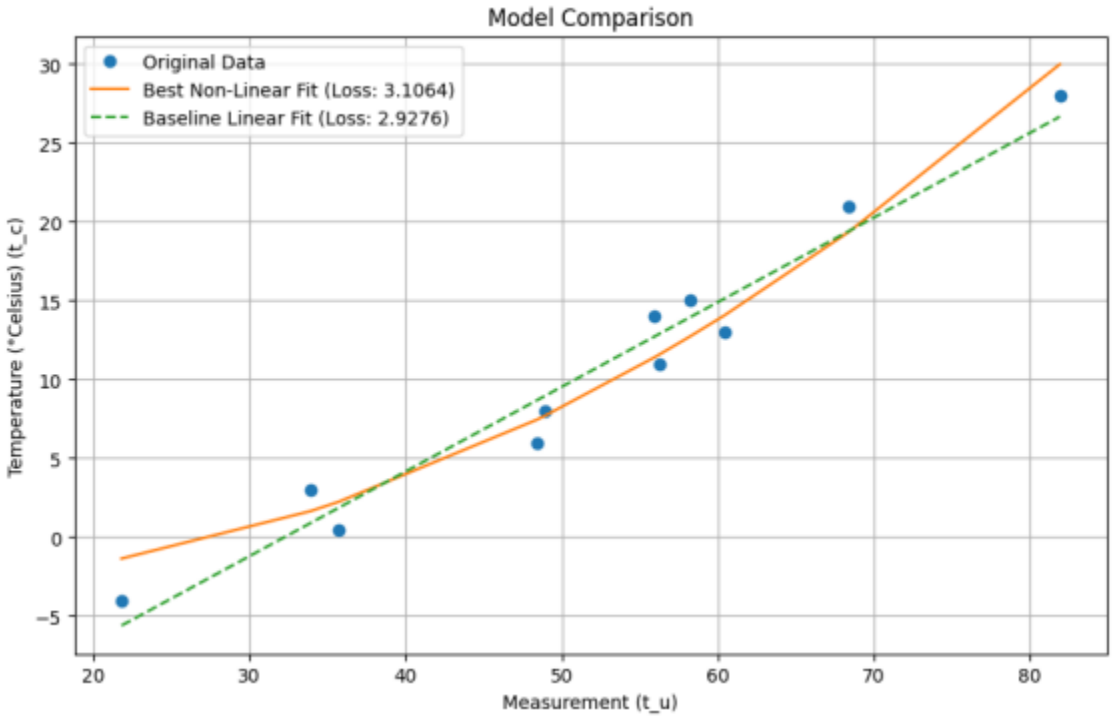
ECGR-4105-001

Joshua Foster, 801268118

Repo Link: https://github.com/jfoste81/ECGR4105_Homework/tree/main

Problem 1:

I found the best learning rate for the non-linear model to be .1, with a loss of only 3.1064. The only other learning rate to come close to this was .01, which had a loss of 3.8327. After re-running the linear model example given in the slides, I found that the non-linear model is slightly worse on the data than the linear model example. To showcase this, I graphed the results for both models in the notebook.



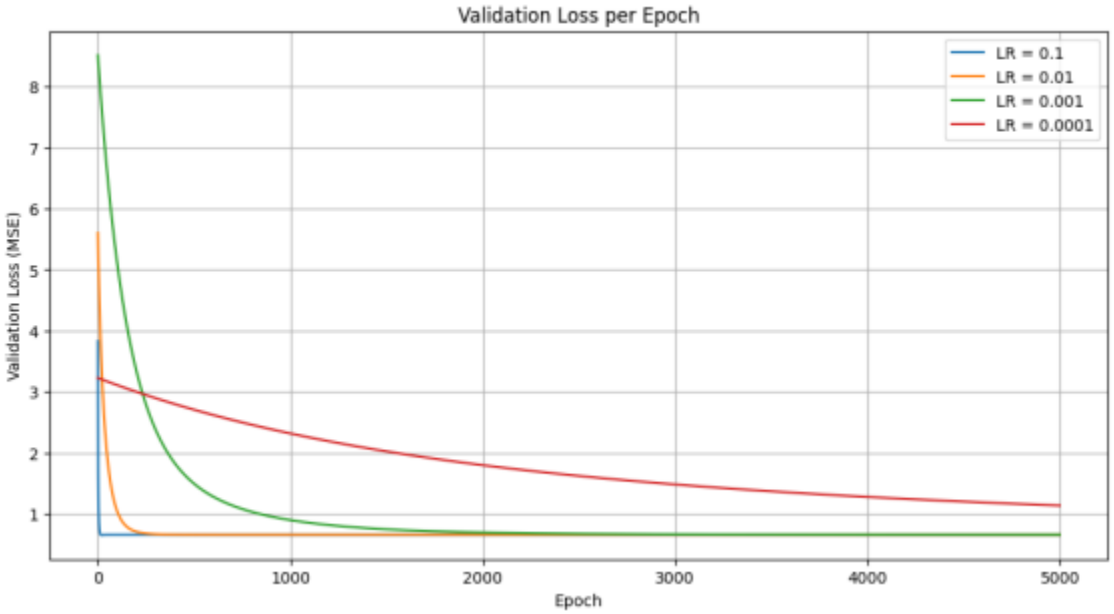
We can see that, while the non-linear model does well to generalize to the data, it doesn't outperform the linear model's predictions. This is also showcased well when comparing the loss, where the non-linear has a loss of 3.1064, while the linear model has a loss of 2.9276. While the difference is slight, it still shows that the linear model learned and generalized better than the non-linear model.

Problem 2:

a) I found the best parameters to be:

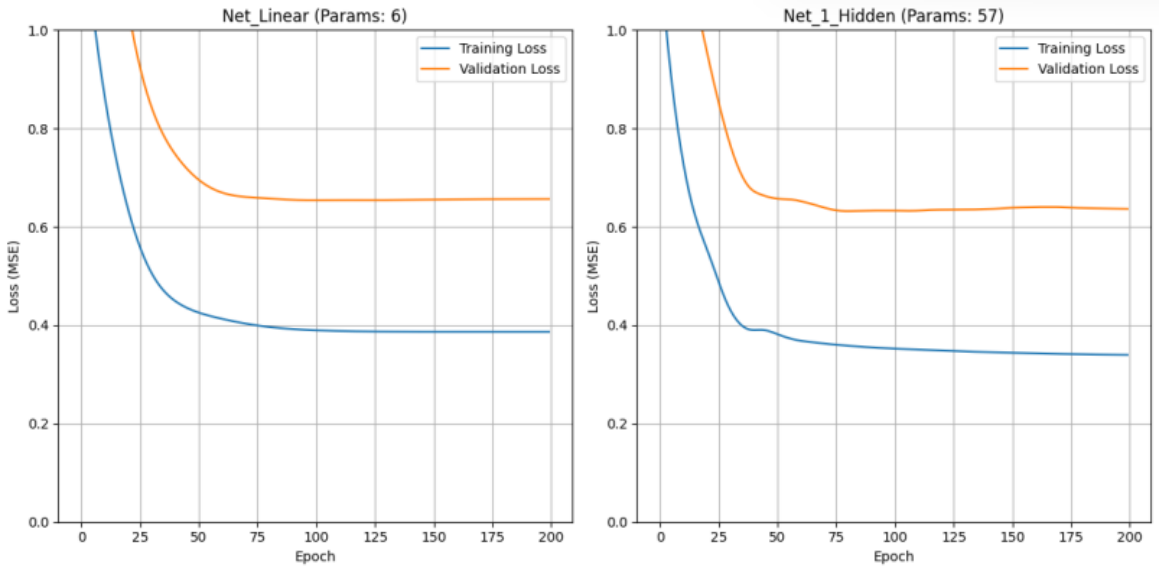
1. W1 (‘area’): 0.35835692
2. W2 (‘bedrooms’): 0.05968103
3. W3 (‘bathrooms’): 0.31853154
4. W4 (‘stories’): 0.2296232
5. W5 (‘parking’): 0.15553762
6. B (bias): -0.0119

b) Using these parameters, I found the best learning rate to be 0.1 with a final loss of .6565
   1. I graphed the results between 4 different learning rates (0.1, 0.01, 0.001, 0.0001) in my notebook. We can see how the 0.1 learning rate converges much quicker than the other learning rates (one of the learning rates not even converging in the allotted 5000 epochs!). The only other learning rate to match the .6565 loss was the 0.01 learning rate but took over 500 epochs to reach it compared to the sub-500 epochs that it took the 0.1 learning rate.



Problem 3:

a) For our single hidden layer with 8 nodes, it took .2 seconds to train and finished with a loss of .3396 and an accuracy of .6365. We can compare it to the original linear model, where on 6 parameters it was trained in .14 seconds (on 200 epochs) and finished with a loss of .3866 and an accuracy of .6565. This means that our single hidden layer model performed slightly better on the data with the expansion of trainable parameters, up to 57 from 6. We can see from the graph at the bottom of my notebook that there was some slight uncertainty for this model at first, but then it began to confidently learn and improve its loss and accuracy. This is interesting as the curves for the linear model are much smoother and more gradual, but with our network it was more up and down at the beginning before it learned.



b) For our multi-layer model, we see that it trained on 201 parameters in .33 seconds with a loss of .2741, but an accuracy of .6505. Compared to the single layer model, we see that the loss went down, but the accuracy decreased. This indicates that there is a degree of overfitting occurring in this model, probably due to the increased complexity of the model. We can see this in the graphed comparison of both models, where the multi-layer model's accuracy is much more jagged and lowers slightly but increases at the end. During this, the loss is steadily decreasing. This shows that the model was learning the training data rather than the general trends within the data, which is the cause of the lowered validation accuracy in the later epochs. Overall, we can see that the single layer model from part 3a was the best performing model.