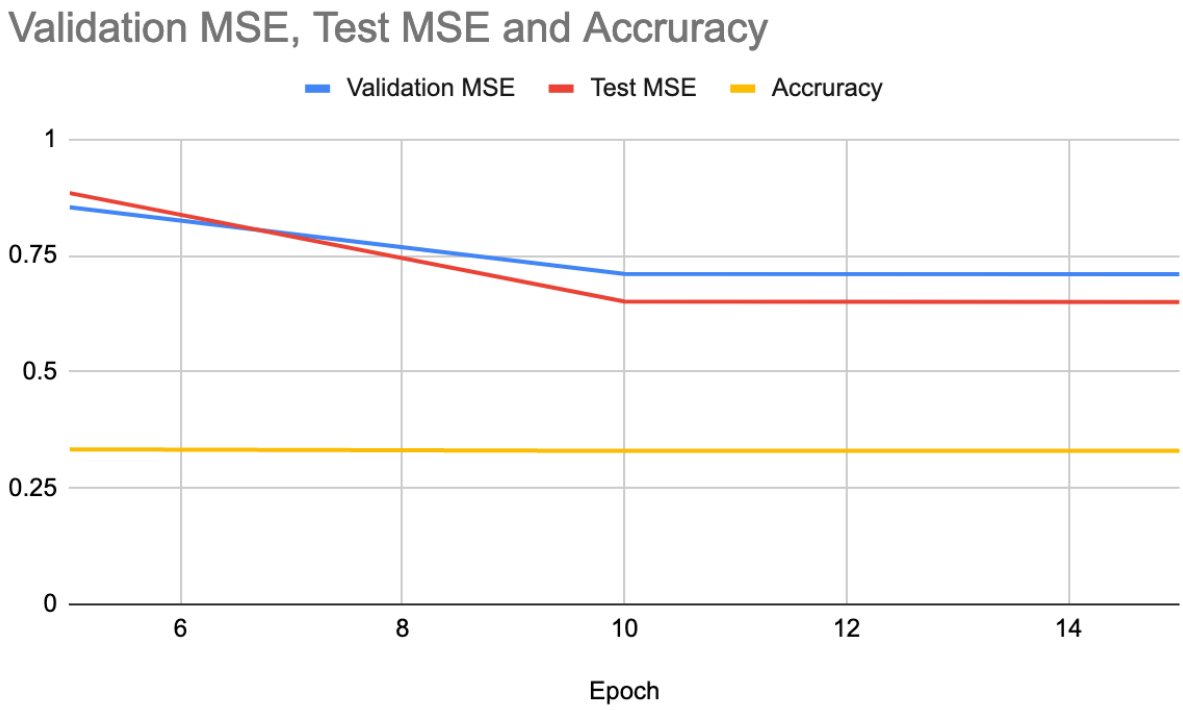
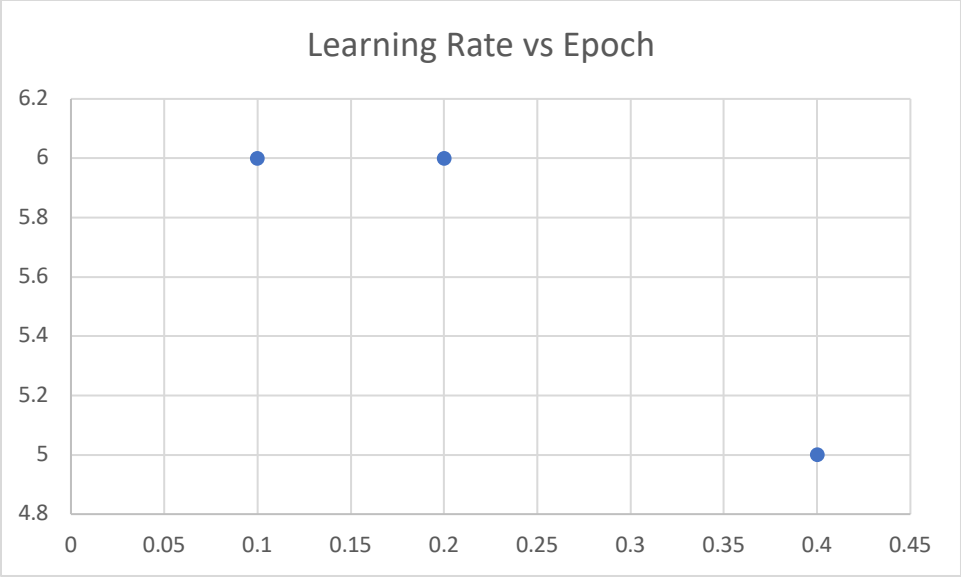
1. I wrote the code and my implementation includes a single hidden layer with arbitrary number of nodes, the ability to randomly initialize weights, stochastic weight updates, a validation set stopping criterion, shuffling, and momentum. Please see the "fit" function and the subsequent function used by fit for all the requirements. The momentum term is built into the formula, setting momentum to zero disables momentum. While running my model on the debug and eval set, I realized that 10 epochs is not enough to properly train the model. So, after 10 epochs, I still had low accuracy. However, when I randomly initialized the weights, I was able to hit 100% accuracy after 5 epochs in the eval data set. I think it is important to not initialized your weight to zero, but instead to randomly initialize the weights for maximum effectiveness.



2. It seems like the number of epoch does not decrease the loss, nor does it increase the accuracy.
3. The Vowel data set is harder because it has more output. The vowel data set has 10 different output classes as opposed to 3 from the iris data set. The vowel dataset also has more input, which leads to a more complex neural net. The baseline accuracy for my model is 65% after 10 epochs.

   I used the test/train features because it seems like the most like the data set, I've built my model to handle. Also splitting by speaker diluted the data too much and was subject to overfitting. Splitting by gender, I felt might introduce a bias that would skew the results.



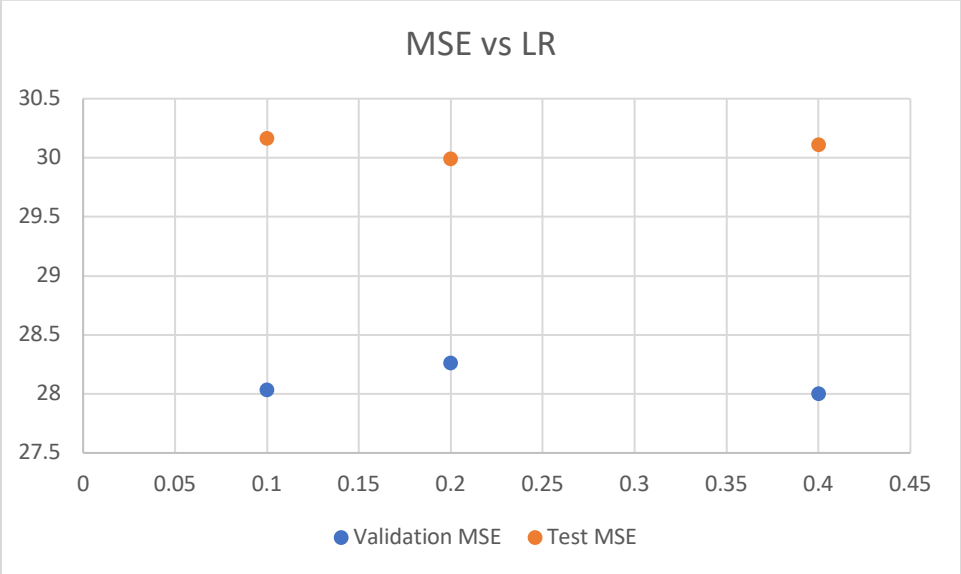
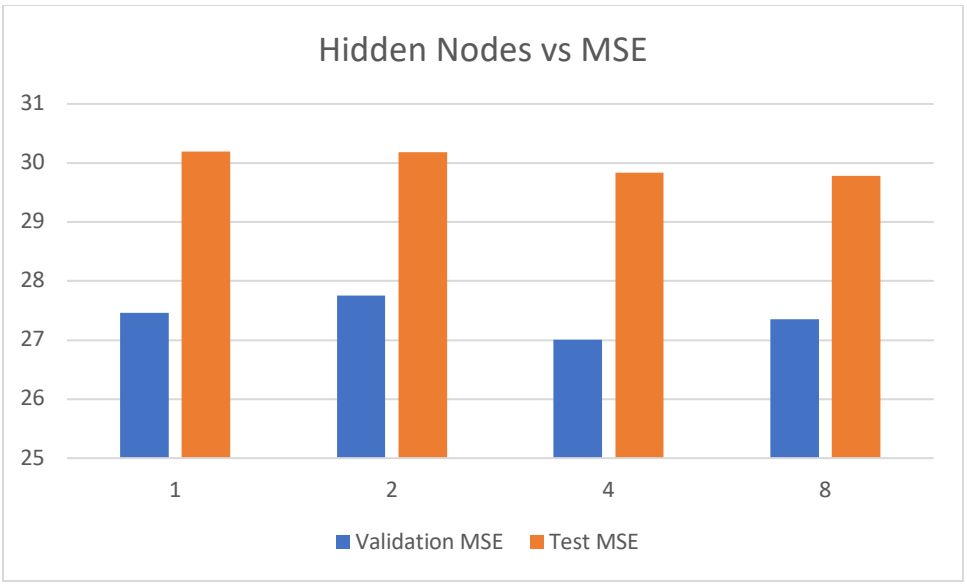
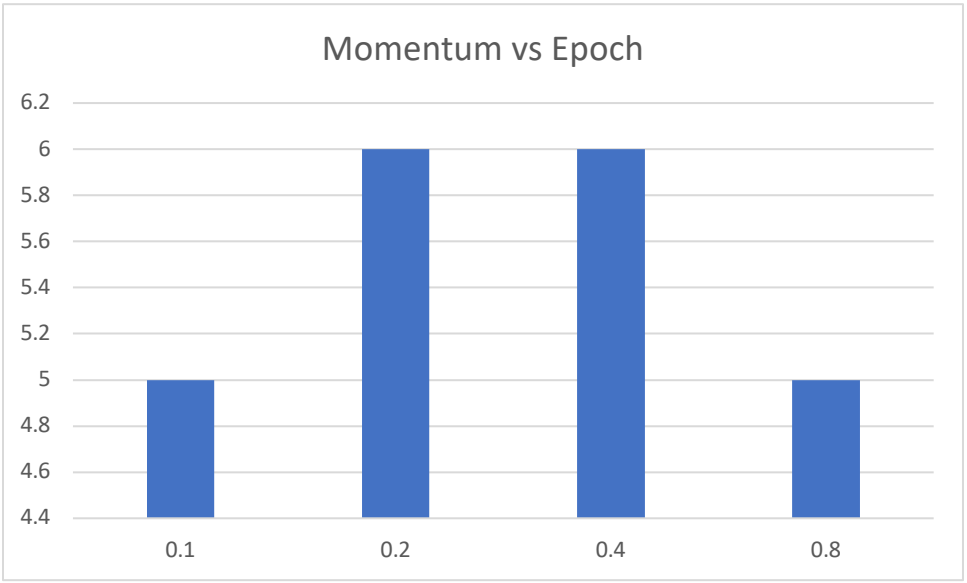


Figure 1: MSE vs Learning Rate

The best learning rate seems to be around 0.5, it has the best mix of speed and lowest MSE loss.

4. 

It seems like fewer node led to higher accuracy, from 1 through 4 nodes, the accuracy hovered around 80%. When I increased the node count to 8, it dropped to 73%. However, the loss decreased as I increased the number of nodes. It seems like 4 hidden nodes is the best for vowel data set according to accuracy and MSE loss. I think fewer nodes led to higher accuracy because, increasing the number of nodes increases graph complexity. The gradient descent calculation is pretty bare bones and minimal and might be able to backpropagation the error well enough in a complex graph.

5. 

It seems like increase momentum didn't really decrease the number of epochs it took to train the model. My validator stopping criterion is based on MSE loss for each batch, and so many things could make the MSE loss increase and therefore stop the training too early. I think my stopping criteria is too sensitive, which is why momentum isn't having an effect on the training time.

6. I used the magic04.csv data set for this problem. The dataset about predicting the type of star using the shape of the object. The SK version of the MLP classifier beats my model every time, on everything. It has lower loss and higher accuracy. Increasing the number of hidden nodes and layer, increases accuracy. The best activation function is the Relu, their default activation function beats my sigmoid activation function. The learning rate peaks at around 0.001. Regularization the data leads to better results, usually. The default momentum good, increasing it too much leads to higher loss. Early stopping will cause worse result than letting the model finishing training. I achieved 81% accuracy with the default parameters, solver = ‘adam’, 2 hidden layers each with 7 nodes, and alpha = 0.001