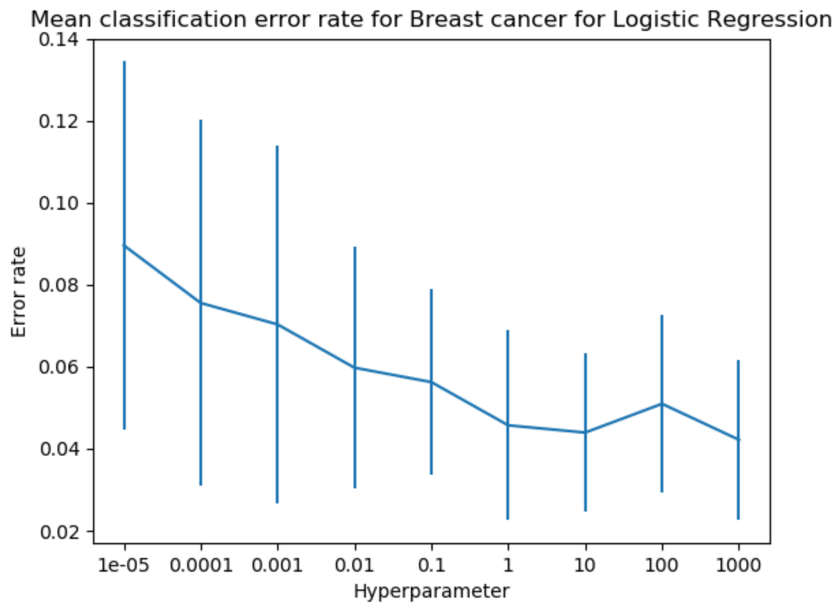
Emily Moeller

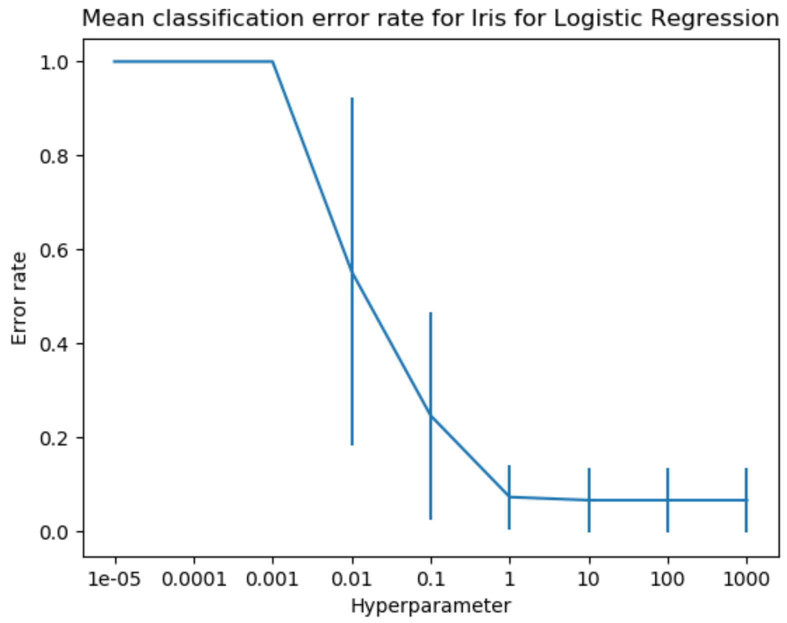
11/26/19

Homework 4

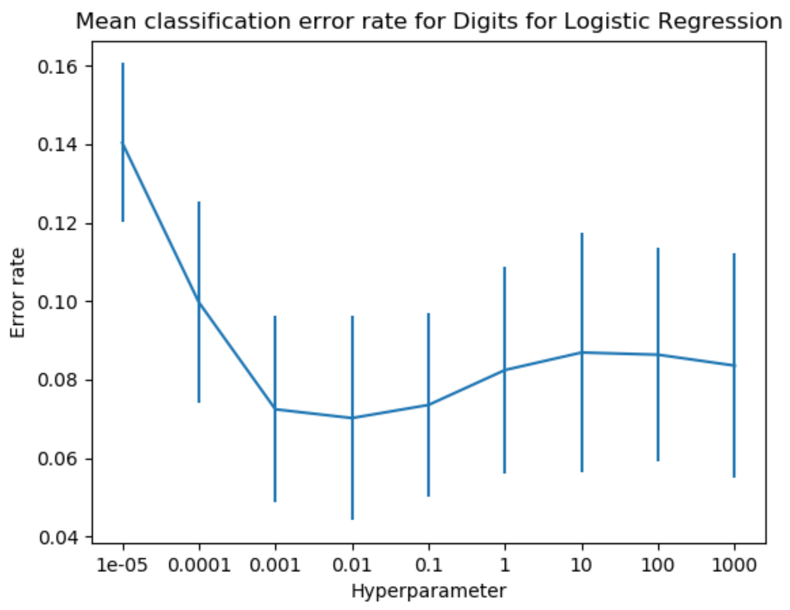
Logistic Regression



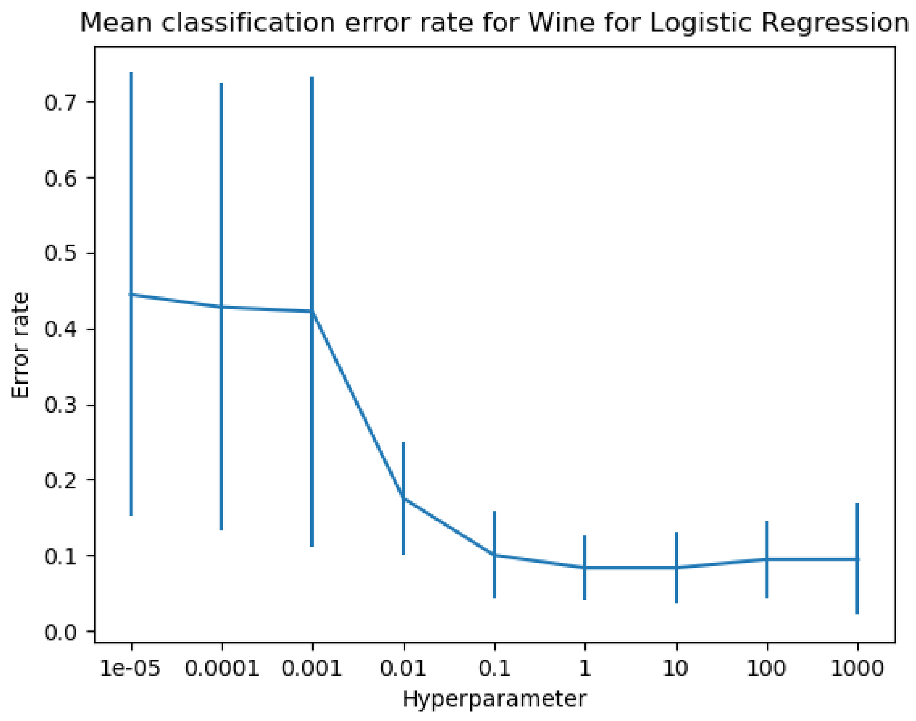
I would choose C=100 because it has the lowest mean error rate and the lowest upper and lower bounds of the standard deviation of the error rate.



I would choose C=1 because it has the lowest mean error rate and the lowest upper and lower bounds of the standard deviation of the error rate. It is also the value right before the function converges, so the algorithm will be fast and it will converge.

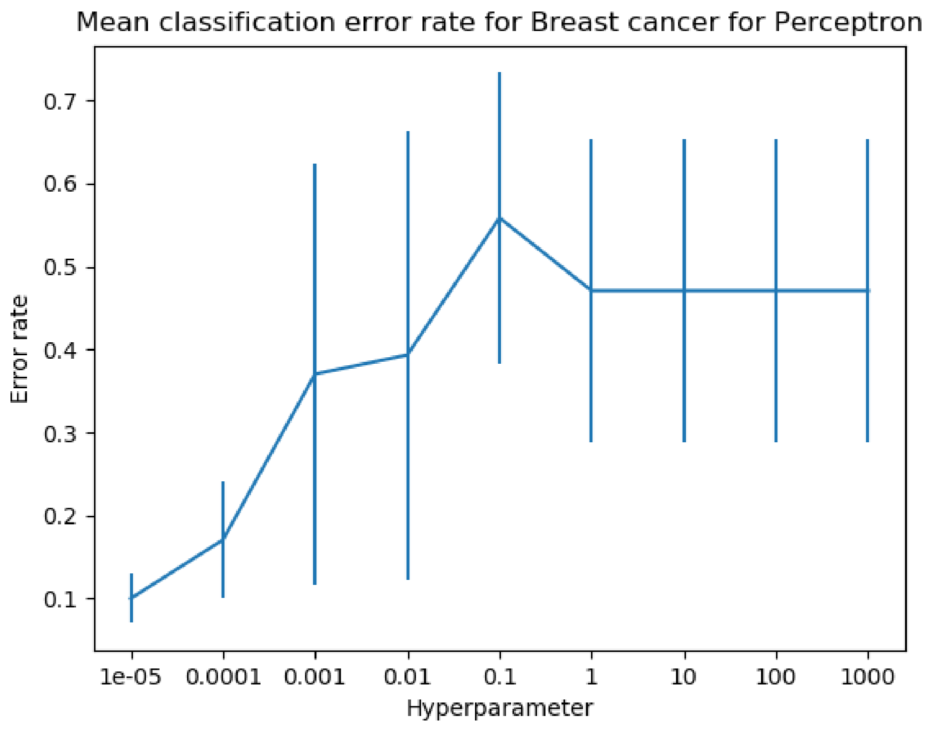


I would choose C=.01 because it has the lowest mean error rate and the lowest upper and lower bounds of the standard deviation of the error rate.

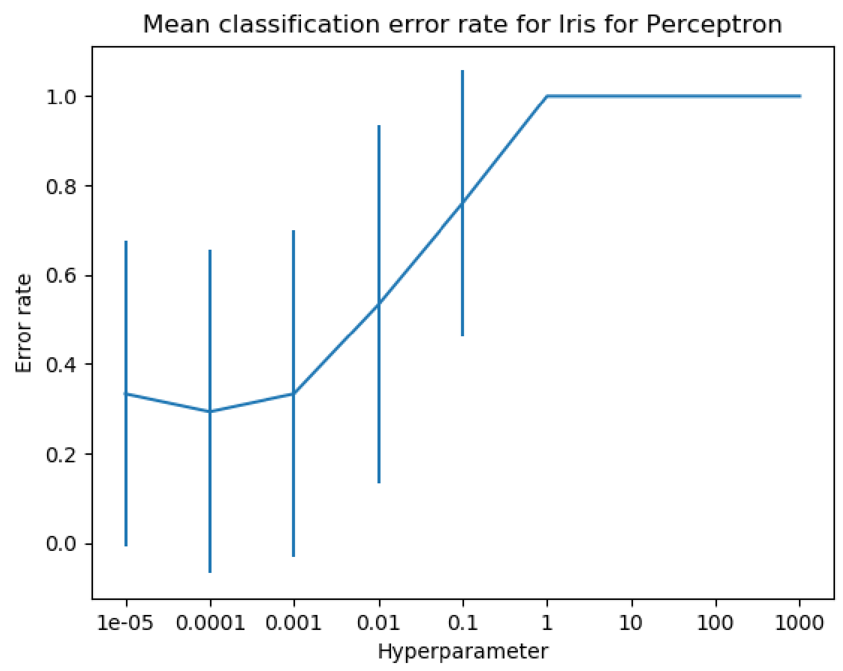


Using a conservative approach, I would choose C=1 because it has the lowest mean error rate and the lowest upper bound of the standard deviation of the error rate. This way, in the worst case scenario, it will yield the best results of all the hyperparameters. It is also the value right before the function converges, so the algorithm will be fast and it will converge.

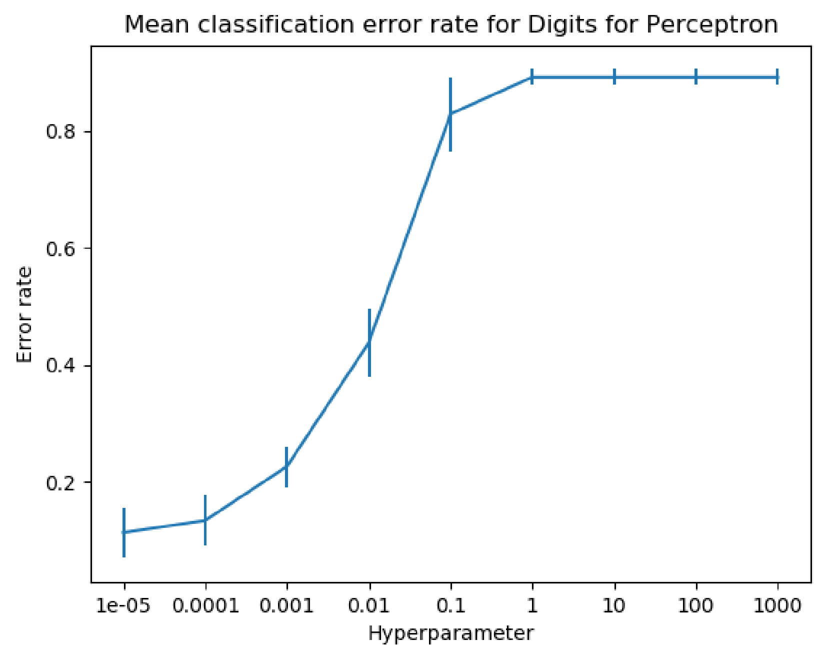
Perceptron- just a warning



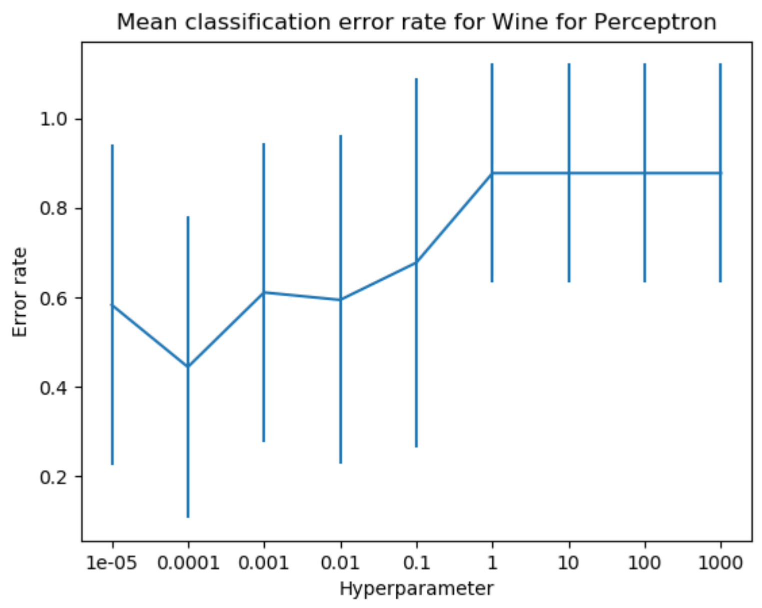
I would choose alpha=1e-05 because it has the lowest mean error rate and the lowest upper and lower bounds of the standard deviation of the error rate.



I would choose alpha=0.0001 because it has the lowest mean error rate and the lowest upper and lower bounds of the standard deviation of the error rate.

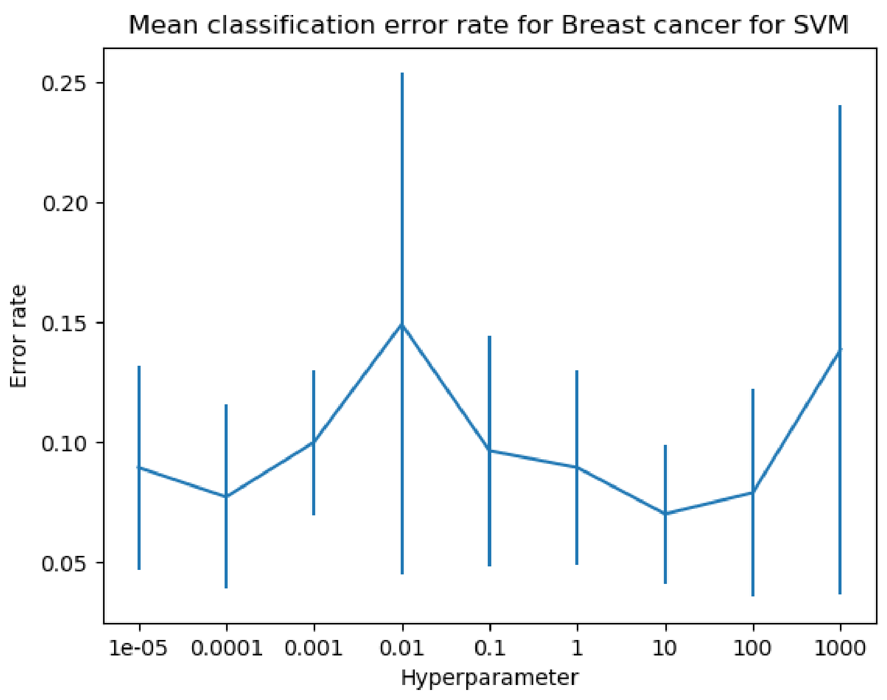


I would choose alpha=1e-05 because it has the lowest mean error rate and the lowest upper and lower bounds of the standard deviation of the error rate.

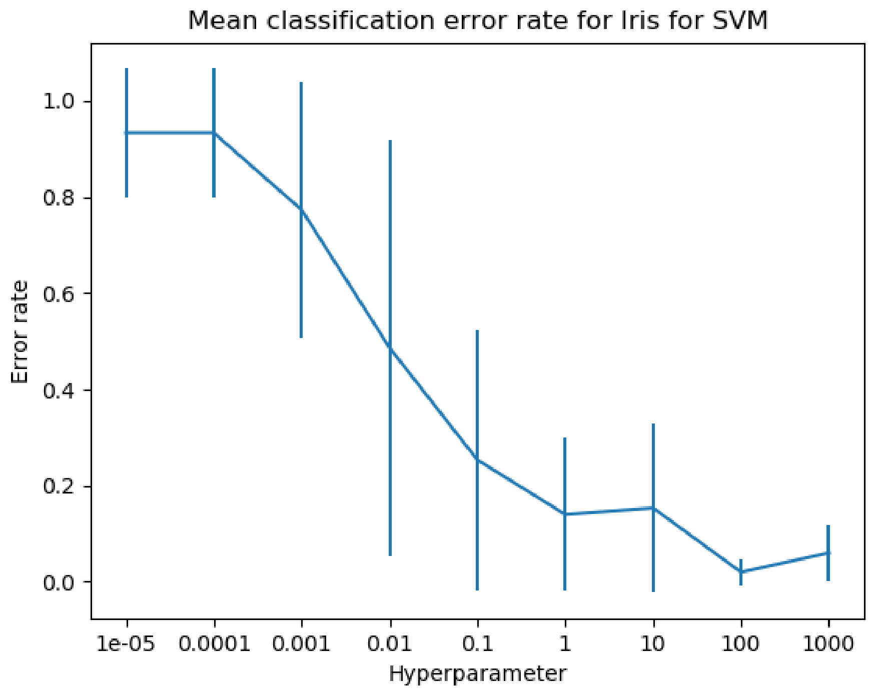


I would choose alpha=0.0001 because it has the lowest mean error rate and the lowest upper and lower bounds of the standard deviation of the error rate.

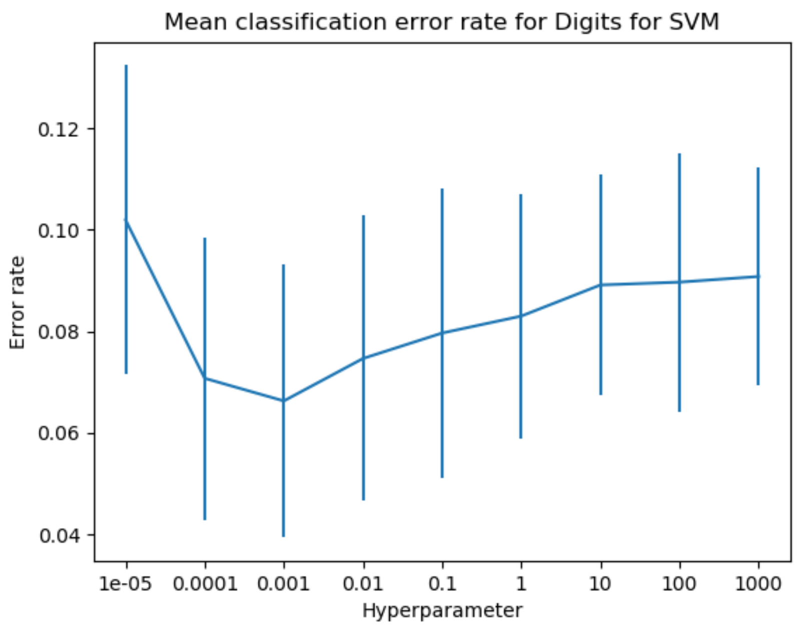
<u>SVM-</u>



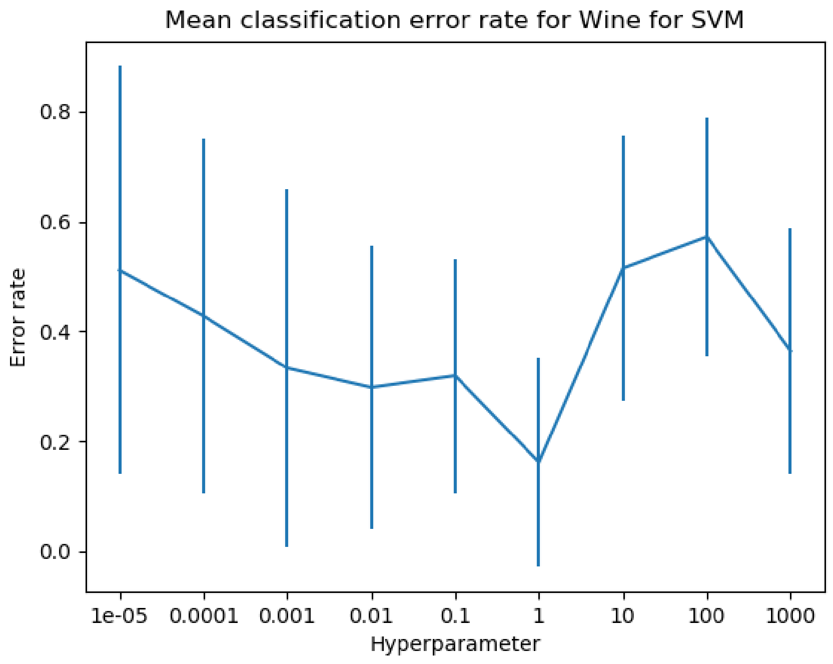
Using a conservative approach, I would choose C=10 because it has the lowest mean error rate and the lowest upper bound of the standard deviation of the error rate. This way, in the worst case scenario, it will yield the best results of all the hyperparameters.



Using a conservative approach, I would choose C=100 because it has the lowest mean error rate and the lowest upper bound of the standard deviation of the error rate. This way, in the worst case scenario, it will yield the best results of all the hyperparameters.

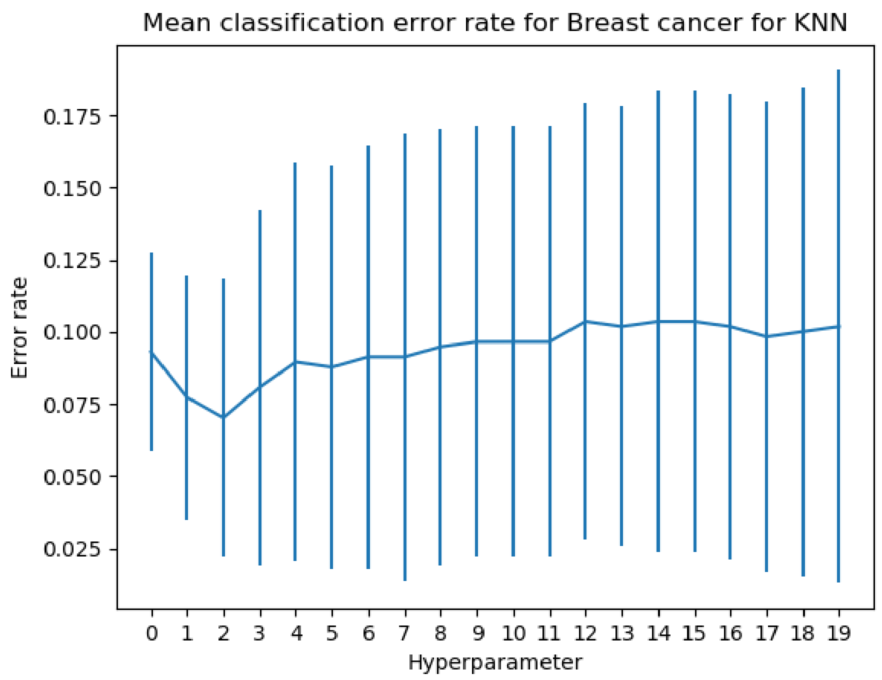


Using a conservative approach, I would choose C=0.001 because it has the lowest mean error rate and the lowest upper bound of the standard deviation of the error rate. This way, in the worst case scenario, it will yield the best results of all the hyperparameters.

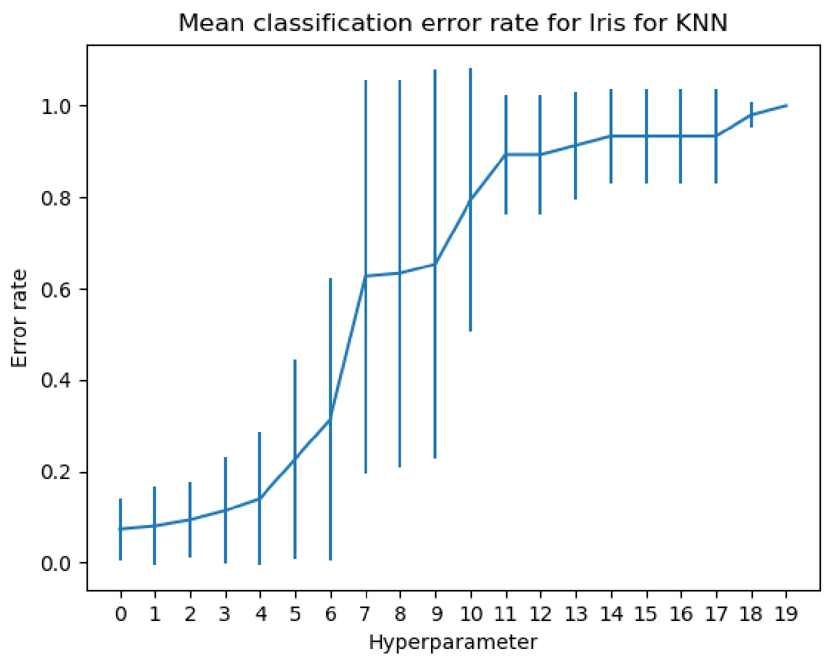


I would choose C=1 because it has the lowest mean error rate and the lowest upper and lower bounds of the standard deviation of the error rate.

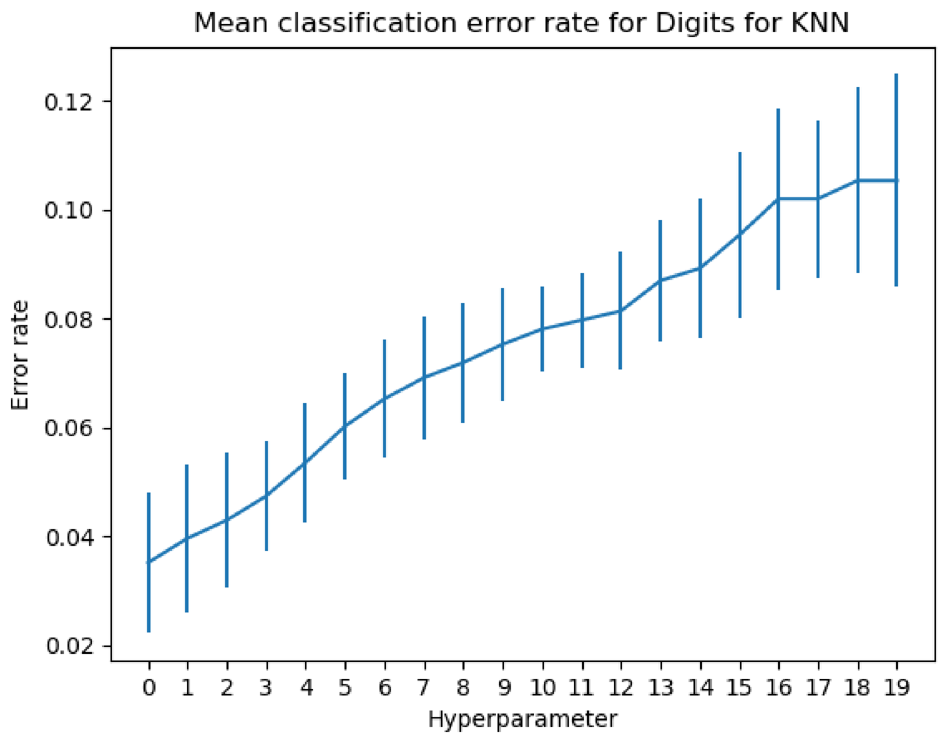
KNN



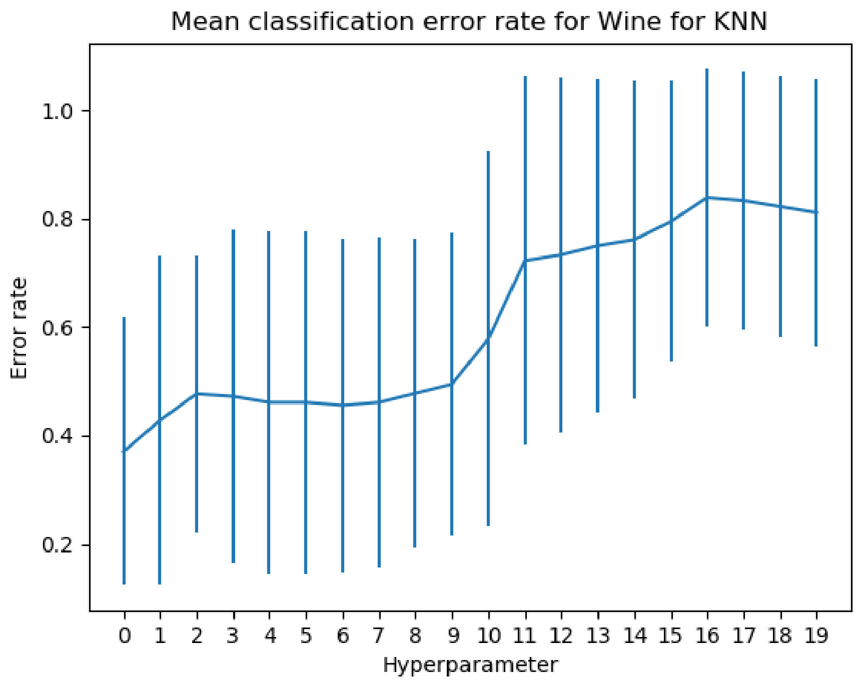
Using a conservative approach, I would choose x=2 because it has the lowest mean error rate and the lowest upper bound of the standard deviation of the error rate. This way, in the worst case scenario, it will yield the best results of all the hyperparameters.



Using a conservative approach, I would choose x=0 because it has the lowest mean error rate and the lowest upper bound of the standard deviation of the error rate. This way, in the worst case scenario, it will yield the best results of all the hyperparameters.



I would choose x=0 because it has the lowest mean error rate and the lowest upper and lower bounds of the standard deviation of the error rate.



I would choose x=0 because it has the lowest mean error rate and the lowest upper and lower bounds of the standard deviation of the error rate.

Things to do:

- Don't think you shuffled code in cross validation
- Problem 3
- Test one more time on lab machines?

1. When training neural networks, we often use the following error function Error = (correct−output)^2. Give two specific reasons (along with explanations) why the above error function is preferred over the following error function: Error = |correct−output|.

Squaring the difference between the correct values and the predicted values emphasizes greater differences in the error, while the absolute value function simply computes the absolute error. This way nodes that are less useful will be given a smaller weight and punished more significantly than they would be if the absolute error was used. Therefore, those nodes will not be as significant in the final training of the model.

Another reason is that the squared error function is differential, meaning that you can perform optimization by gradient descent. This means that you first calculate the gradients of the loss function, and then update the existing parameters in response to the gradients. This continues until a local minimum is reached. Gradient descent plays an important role in speeding up convergence and attaining lower error rates.

2. The neural networks we discussed in class had every layer connected to only the next layer. Suppose we were to modify this to allow either: (i) layers can connect to future layers (left figure) or (ii) layers can connect back to previous layers (right figure). Which of these modifications is more difficult to adapt our learning method (gradient descent) to fit? Explain your reasoning (a general statement) and provide a concrete example to back up your thoughts (a specific example).

I think layers connecting back to future layers is more difficult to adapt to gradient descent, because it would require the network to store all computations from all previous layers, just in case they were to be used in a future layer. This could become more complicated because now nodes that never had connections to each other must be able to be connected. Meanwhile, in the case of layers that can connect back to previous layers, it would require a lot less work because a network already has established connections to the previous layer. In fact, this is the premise that is used for back propagation for gradient descent, which essentially backpropagates the error from the output layer to help update the weights in the hidden layer.