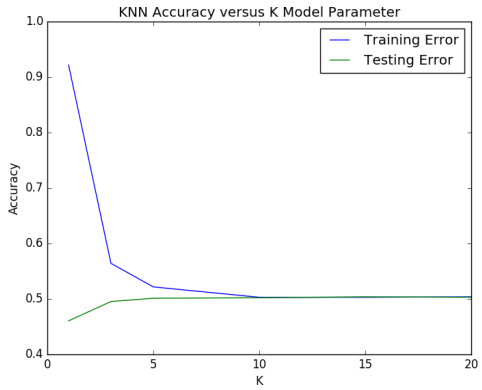
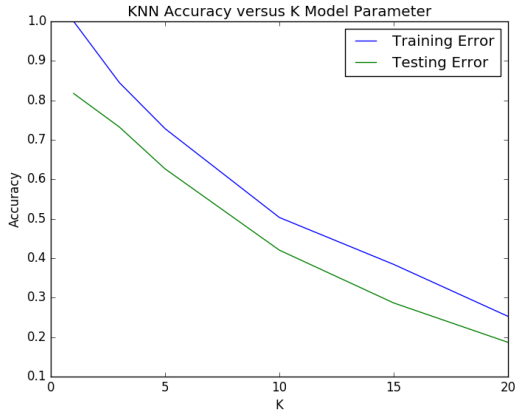
Alec Chapman
Applied Machine Learning
Spring 2017
HW4

A. I would say that the best parameter is $K = 3$. The KNN algorithm works best with fewer neighbors. On the MNIST dataset, the algorithm performed best in both testing and training with just one neighbor. On the 20NG, the testing performed best with 3 neighbors, while the training performed best with just 1. When there are fewer neighbors to compare to, these neighbors are more likely to be of the correct class. As K grows, more and more instances are included in the comparison and they are less likely to be of the same class. Eventually it becomes a question of which class is more prevalent.

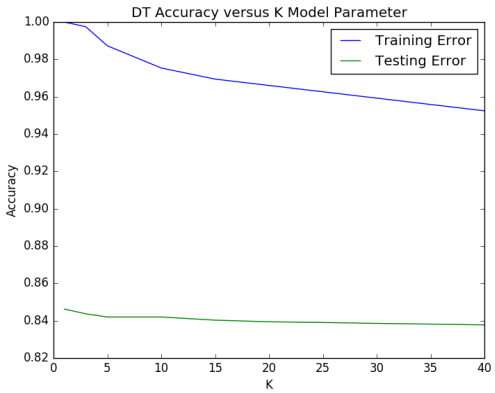


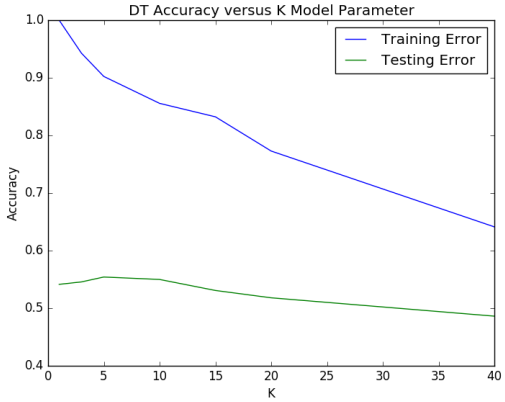


20NG **MNIST**

B. The best parameter for the decision tree looks to be about $K = 5$. The training data does consistently better with a higher K number, but that can lead to overfitting. With a smaller K number, the decision tree is deeper and will have split based on more features, and there the nodes will be purer, making it more exact.

20NG **MNIST**





C. The Decision Tree overfits in the MNIST set. Although the performance worsens in both the training and testing, there is a consistently sizable gap between the two performances. The gap in the 20NG set is smaller and the DT performs pretty well.
D. The KNN underfits in the 20NG set. Both the training and testing accuracies are very low. It does not appear to be over- or underfitting in the MNIST as the two performances are pretty consistent and achieve good results with a lower parameter.
E. I would choose the KNN algorithm with a *K* value of 1 for the MNIST classification task. With this parameter, the algorithm got a testing accuracy of ~ 0.8. The DT did much more poorly.
F. For the 20NG task I would use a decision tree with a *K* value of 1. Testing maintained an accuracy of ~ 0.8, while the KNN barely broke 0.5.