Summary Report

Data: This data set is called the abalone data set. Abalone's are a form of sea snails whose age can be determined through a "boring and time consuming" process by boring into the shell and counting the rings much like a tree. In order to save our marine biologist friends' time, we will attempt to predict the number of rings in the shell by measuring:

1. Sex-a factor variable with three levels "M" for adult male, "F" for adult female and "I" for infant.
2. Length- a measurement of the longest distance from one tip to the other lengthwise on the snail (mm)
3. Diameter- a measurement similar to the length with the exception that it must be perpendicular to the length (mm)
4. Height - measurement of height with snail in shell (mm)
5. Whole Weight - measurement of the complete snail (gram)
6. Shucked Weight - weight of meat (g)
7. Viscera weight - weight of gut after being bled (g)
8. Shell weight - weight of dried shell after snail has been removed(g)

Pre-processing: The data, being created by scientists for predictive purposes, is pretty clean. There are 1307, 1342 and 1528 in the "F", "I" and "M" category respectively. According to the Shapiro-Wilks test, none of the continuous variable are normal and will thus have to discretized for Naive Bayes.

Beginning: Here I make a function that calculates the Root Mean Square Error (RMSE) and the Mean Absolute Error (MAE) and returns a 1x2 matrix with those values. I also create a master test and training split. I do this because I want to be able to more accurately compare the first build for all the models and need master test and train sets to do this. Also, for a general note, sometimes the parameters that minimize RMSE and MAE are different, so I built the algorithms to cross-validate for both sets of parameters. However, those parameters do not vary much from each other and so the results for both should be similar but I added them for the sake of completeness. I also cross validated using both 10-fold cross validation and Delete-D because the Kohavi paper shows that 10-fold cross, while stable, is pessimistic and adding a 25 repetition, delete-d cross-validation along with a 10-fold cross might give a better idea of where the true accuracy really is. Finally, I use the tune function multiple times during the paper and it has its own cross-validation, generally 10-fold, and reports the Mean Square Error. This is why some of the time I only create one cross-validation function and why

Artificial Neural Networks: For artificial neural networks, I first created a for-loop to test models with different numbers of hidden nodes in one layer so I can see which provides the most accuracy. I chose the range 1-15 for the number of nodes because having more than that seemed overly complex. The best accuracy was found with a 3-node hidden layer. Here are the results:

$$\begin{array}{cc} Delete-D\ \ cross\ \ validation & 10-fold\ \ cross \\ RMSE = 2.077799;\ MAE = 1.466413; & RMSE = 2.217157;\ MAE = 1.640443 \end{array}$$

Decision Trees/Random Forest: For decision trees, I first used the straight `rpart` command to get an idea what type of accuracy that can be expected from one tree. Then, I used the tune command to build the best random forest I could. Since tune includes a 10-fold cross validation, I only needed to add a delete-d cross-validation. The tune function provides only the Mean Square Error on the OOB data, so it does agree with the Delete-D error. Here are the results for the forest:

$$\begin{array}{cc} Delete-D\ \ cross\ \ validation & 10-fold\ \ cross \\ RMSE = 2.134893;\ MAE = 1.513462; & 4.709902 \end{array}$$

Support Vector Machines: In order to find the best results for support vector machines, I created a for-loop that would run the tune command for a set range, find the best parameters and then build a new parameter range based on that. Since it repeats five times, it should find a good guess as to the best parameters. Here are the results:

$$\begin{array}{cc} Delete-D\ \ cross\ \ validation & 10-fold\ \ cross \\ RMSE = 2.252857;\ MAE = 1.535832; & 5.124983 \end{array}$$

Naive Bayes: The only parameter to control here is the number of levels to discretize input values into. I tried values between 2 and 15 because having one level is ridiculous and, due to measurement error, I thought splitting into 16 levels would be too fine. Also, to find the best results, I had to factorize the ring value along with discretizing the input variables. Here are the results:

$$\begin{array}{cc} Delete-D\ \ cross\ \ validation & 10-fold\ \ cross \\ RMSE = 3.236258;\ MAE = 2.070876; & RMSE = 3.457027;\ MAE = 2.768922 \end{array}$$

KNN: For KNN, I had to create two dummy variables for the "M" and "F" factor in the sex variable. The only parameter to control here was the

number of neighbors that k had to consider. I tested all the values between one and fifty; kept the ones that minimized both RMSE and MAE; and cross validated them. Here are the results:

$$\begin{array}{cc} Delete-D \quad cross \quad validation & 10-fold \quad cross \\ RMSE = 2.941849;\ MAE = 2.056951; & RMSE = 2.769610;\ MAE = 2.001218 \end{array}$$

Weighted KNN: Since I discovered in a previous homework that the "optimal" kernel does not always return the best values, I tested over all of the kernels available and k values between 1 and 50. Since the command I used to find those uses a leave-one-out cross validation, I only had to make a 10 fold cross validation. Here are the results:

$$\begin{array}{cc} Leave-one-out \ \ cross \ \ validation & 10-fold \quad cross \\ RMSE = 1.519241;\ MAE = 4.843951 & ;RMSE = 2.206253;\ MAE = 1.526947 \end{array}$$

Conclusion: I would say that the Artificial Neural Network would be the best in this case. It had the best results and seemed to be the most consist with its error. Also, since it is an eager learner, I would argue that it would easier to hand it someone else who has no experience with programming to implement because they could theoretically just follow the path or put it into Excel.