**Homework 10 – Classification with Support Vector Machines**

Support vector machines (SVM) are a highly flexible and powerful method of doing supervised machine learning. Supervised learning means that there is a criterion one is trying to predict. The typical strategy is to divide data into a training set and a test set (for example, two-thirds training and one-third test), train the model on the training set, and then see how well the model does on the test set.

In this homework, we will use a chunk of the “diamonds” dataset from the ggplot2 package to do some classification with an SVM. For the sake of simplicity, we will see if we can correctly classify the “cut” of a diamond into one of two categories “Premium” or “Ideal.”

**Part A: Load and condition the data**

1. The diamonds data set is in the ggplot2 package, so make sure you library( ) that package. While you’re at it, you will also need to install( ) and library( ) kernlab.
2. There are five cuts of diamonds: Fair, Good, Very Good, Premium, and Ideal. To simplify our classification task we will focus only on Premium and Ideal. The following expression selects the data that fits those two categories:  
   *diamonds[(diamonds$cut=="Premium" | diamonds$cut=="Ideal"), ]*  
   Assign the result of that expression to a new data frame and use that data frame for subsequent work.
3. The clarity and color variables in the data frame are “ordered factors.” That means that for some analytical purposes (like this one) you can convert the factor level directly into a number and it will make sense. Use *as.numeric( )*  to accomplish that.
4. Write a block comment describing the meaning of each variable in the data frame.

**Part 2: Create training and test data sets**

Using techniques discussed in class, create two datasets – one for training and one for testing.

1. Pages 235 - 237 of the book describe how to create a training data set and a test data set. Following the strategy in the book, the training data should contain about two thirds of the whole data set, with the remaining one third going to the test data.
2. Use the *dim( )* function to demonstrate that the resulting training data set and test data set contain the appropriate number of cases.

**Step 3: Build a Model using *ksvm( )***

1. Build a support vector model using the *ksvm( )* function using all of the variables to predict cut. Once you have specified the model statement and the name of the training data set, you can use the same parameters as shown on page 237:   
   *kernel= "rbfdot", kpar = "automatic", C = 5, cross = 3, prob.model = TRUE*
2. Write a block comment that summarizes what you learned from the book about those parameters. The two parameters of greatest interest are *C=5* and *cross=3*.
3. Store the output of the kvsm( ) run in a variable and then echo that variable to the console. You will know that you are on the right track if your cross-validation error is reported in the neighborhood of 0.08. The other output information is mainly diagnostic and is not of great concern at this time.

**Part 4: Predict Values in the Test Data and Create a Confusion Matrix**

1. Use the *predict( )* function to validate the model against test data. Assuming that you put the output from the *ksvm( )* call into svmOutput and that your test data set is in a data frame called testData, the call would be:  
   *svmPred <- predict(svmOutput, testData, type = "votes")*
2. Now the svmPred object contains a list of votes in each of its rows. The votes are either for “Premium” or “Ideal”. Review the contents of svmPred using *str( )* and *head( )*.
3. Create a confusion matrix (a 2 x 2 table) that compares the second row of svmPred to the contents of testData$cut.
4. Calculate an error rate based on what you see in the confusion matrix. See pages 243-244 for more information.

**Expert Mode: Explain, in a block comment, why it is valuable to have a “test” dataset that is separate from a “training” dataset?**

**Ultra-Power Expert Mode: Use lm( ) to reproduce what you did above with ksvm( )**