Visualization and Learning Structure: Problem Session 7/11/16

We'll be looking at a famous banknote authentication data set. Images were taken of 1372 banknotes, some counterfeit and some genuine. Wavelet transformation tools were used to extract the following descriptive features of the images: *Variance*, *Skewness*, *Kurtosis*, and *Entropy*. We also have the true label for whether or not the banknote is genuine (Yes = 1, No = 0).

Our goal is to build a classifier predicting whether a banknote is real.

1. Download the banknote data from our website and read into R.

   `banknote<-read.table("banknote.txt",header=T)`

   Use `attach(banknote)` to create a variable for each column.

For these problems, we're providing example code using the `tree` library/function. You could also use the `rpart` library/function. See `help(tree)` and `help(rpart)` for differences in syntax.

2. Build a classification tree predicting Class from the four descriptive variables.
   You'll need to make sure that R knows Class is a categorical variable:

   `banktree<-tree(as.factor(Class)~Variance+Skewness+Kurtosis+Entropy,split="gini")`

   Graph the tree and its labels (`plot(banktree); text(banktree)`).

   How many final nodes/leaves do you have?
   How many are predicted to be genuine banknotes?

3. Look at the detailed tree results: `banktree`.

   Are there any terminal nodes (leaves) that are 100% genuine or counterfeit?
   Which terminal nodes (leaves) are close to a 50-50 mix?

4. Now we'll predict the class for all 1372 of our banknotes.
   `predict.class<-predict(banktree, type="class")`.

   We can compare our predictions to the real Class values by tabling the two vectors: `table(Class, predict.class)`

   What is our misclassification rate?

5. Our first classification tree is probably unnecessarily complicated.
   We can prune the tree back to something more manageable, say, 10 nodes:
   `banktree.small<-prune.tree(banktree,best=10)`

- Graph your smaller tree; describe its set of terminal nodes/leaves.
- Now use the smaller tree to predict the class for our banknotes. Use `table()` to compare the predictions to the real class labels. Did our misclassification rate increase or decrease? Why?

6. Often we address over fitting concerns by dividing our data into a training data set and a test data set. We build the classifier on the training data and then assess its accuracy on the test data set. We would expect the classifier to do well for the training data set (since it was used to build the model), but a robust, stable classifier that didn't overfit should also do fairly well on a similarly generated test data set.

   - Split your banknotes randomly into a training data set and a test data set
     ```
     which.train<-sample(seq(1,1372),686)
     bank.train<-banknote[which.train,]
     bank.test<-banknote[-which.train,]
     ```
   - Now we'll build a tree using just the training data:
     ```
     banktraintree<-tree(as.factor(Class)~Variance+Skewness+Kurtosis+Entropy,
     split="gini",data=bank.train)
     ```
   - Then predict the classes for the test data set:
     ```
     predicted.testclass<-predict(banktraintree,newdata=list(Variance=
     bank.test$Variance,Skewness=bank.test$Skewness,Kurtosis=bank.test$Kurtosis,
     Entropy=bank.test$Entropy),type="class")
     ```

     We can compare the predictions to the true classes in the test data set: `table(bank.test$class, predicted.testclass)`.

     What is our misclassification rate now?

7. We could also use Random Forests to address any over fitting concerns:

   ```
   library(randomForests)  #need to download/install
   ##using the defaults which includes 500 trees
   bank.rf<-randomForest(as.factor(Class)~Variance+Skewness+Kurtosis+Entropy)
   bank.rf
   ```

   Did we do any better? What's the overall random forest misclassification rate?

   Note that random forests here would select two of our four predictor variables for each tree. The more useful scenario is when we have a very large number of variables. We could use all four variables in each tree (and still let RF select a random number of obs) by setting the argument `mtry=4`. Did using all four improve your misclassification rate?