Graded Assignment 4
(Due Tue 11/26)

ISOM 674, Fall 2019

In this assignment we are going to explore the behavior of logistic regression, classification trees, bagging, and random forests using the spam dataset. Instead of using the coded data set that I used for the naive Bayes model, we will use the original data (in the file "spambasedata-Orig.csv"). In this file, all of the word and character frequency variables are coded as percents (of the words or characters, respectively) in the e-mail message and the run-length variables are coded as counts. Also, we are going to be using all 57 features in the data set instead of the 8 that I used in the naïve Bayes example. The documentation for the data set can be found at the follow link: http://archive.ics.uci.edu/ml/datasets/Spambase.

To get you started, I have provided a "helper" R script (Assignment4Starter.r) that:

- Reads in the data
- Permutes the data using a permutation loaded in from the R data file SpamdataPermutation.RData
- Splits the data into a training and validation samples. The match the training and validation samples used in the naïve Bayes calculations we did in Excel in class.

Note: We are not using a test sample in this example, but do not conclude that test samples are not important in practice.

Questions

1. Fit a stepwise logistic regression to the spam data using all 57 features. To get you started, in addition to reading in the data and creating the training and validation samples, the R helper script also computes the "small" and "big" formulas you will need to run the stepwise regression and runs the glm() function to estimate the logistic regression for the "small" model.

   Comments:

   - You can use the code in the file "StepwiseDemo.r" from class 7 to remind yourself exactly how stepwise regression works.
   - Don't worry about any warnings from the stepwise regression telling you that some predictions are numerically 0 or 1.
   - Note: The stepwise regression may take some time to run, so be patient.
   - The predict() function works with the output of the stepwise regression function, so you can make predictions in a manner quite similar to what you have done before. For the output from the gml() function, the prediction will be the log odds rather than the prediction probability. To get predict() to give you the predicted

probability, add the argument type="response" to the arguments of the predict function. The predict function for objects returned from gml() actually calls the function predict.glm(), so you can look at the help for that function for more details if you need to.

- I have provided an R script ROCPlot.r which defines a function ROCPlot() that you can use to make the requested ROC plots and to compute the AUC.

Once you have fit the stepwise logistic regression, compute the ROC plot and answer the questions on the Google Form.

2. Find the best tree model that fits the spam data. Note that the file I used in class (BaggingRandomForests.r), basically provides a template. We need to make a few changes, however:
    - In order to do classification, tree() needs the Y-variable to be of the factor data type. Thus we need to change IsSpam to a factor in both the training and validation data sets. The following code should work:

        TrainData$IsSpam ← factor(TrainData$IsSpam)

        Use similar code for the validation data.

    - Just as in the logistic regression, use all 57 features in the model specification. (Also as in the logistic regression, the ultimate model may not use all 57, but we want there to be an opportunity for each of the 57 features to be used.)
    - When doing classification, the tree fitting procedure can split using either the Gini index or the deviance. The deviance should be essentially the log likelihood function, so let's use that. To do so, add the argument method="deviance" to the tree() function call.
    - As in the class example done in class, you will need to step through all of the tree sizes to find the best one. Let's use AUC as the criterion for model performance (see below).
    - When tree() is used for classification (as we are doing here), add the argument type="vector" to the arguments of the predict() function when you make the predictions for the validation data set. This causes predict() to return the probabilities of each class.
    - Since, when doing classification, the tree() and predict() functions anticipate more than two possible classes, the predict() function with the argument type="vector" actually returns a matrix. Iin our case, this matrix will have the probabilities of the class "0" as the first column and the probabilities of the class "1" as the second column. For the ROC plot and AUC calculations, we need just the vector of probabilities that the response variable is 1. This is the second column. So after you compute PHat using the predict function, you will want to do something like:

        PHat ← PHat[,2]

to get just the probabilities that the response is 1 (i.e., the probability that the e-mail is spam).

Note: When used on the output from tree(), the predict function actually called is predict.tree(). Thus, you can find the documentation using the help for predict.tree.

- The ROCPlot function can be used just to return the AUC in the following manner:

```
AUC ← ROCPlot(Phat,Y,Plot=F)$AUC
```

- You are going to have to rewrite the code a bit because AUC is better when bigger, while root MSE is better when smaller.

Answer the questions on the Google form and submit the plot of the AUC against the number of terminal nodes in the tree. This is the graph that shows which model is optimal. Then, also submit the ROC plot for the tree fit using the optimal number of nodes.

3. Next use bagging to fit a model to the spam data. Again, the R script I used in class provides a template. Since the response variable is a factor, the randomForest() function will perform classification. Use 500 trees and do not set maxnodes. Of course, use all 57 features in the model specification. Notes:
   - The predict() function actually calls predict.randomForest().
   - The predict() function behaves in a similar fashion to the predict() function for trees. This time you need type="prob" as an argument in order to return the matrix with the probabilities of each class.
   - As in question 2 above, you will want to select the second column:

```
PHat ← PHat[,2]
```

   - When you do not set maxnodes, trees of maximum depth are fit.

   Answer the questions on the Google form and submit the ROC plot.

4. Finally use random forests to fit a model to the spam data. Again, the R script I used in class provides a template. Since the response variable is a factor, the randomForest() function will perform classification. Use 500 trees, do not set maxnodes, and you can let mtry be the default value.

   Answer the questions on the Google form and submit the ROC plot.

Instructions for Submitting the Plots for Question 1 to 4

In addition to answering the questions on the Google form, you need to turn in the plots requested above. Note that there are 5 plots requested. Turn these plots in as labeled figures in a pdf file named:

HW4-Plots-EmoryNetID.pdf

Please substitute your Emory Net ID for "EmoryNetID" in the file name above.

As I have indicated before, you can make the required pdf file by first making a Word document containing the figures. An easy way to get a plot into Word from RStudio is to pop the plot out of the RStudio IDE into its own window by using the Zoom button. The window can then be copied using Control-Alt-PrtScr and pasted into the Word document with Control-V.