# Homework 3

*PStat 131/231 - Spring 2016*

*Due May 14th, 2016*

**Question 1**

Get the dataset "spambase.dat" from GauchoSpace and read it with the following code:

```
spam<-read.table("~/spambase.dat",header=T,sep="")
spam$y<-factor(spam$y,levels=c(0,1),labels=c("good","spam"))
```

**Data Info** The Data Set was obtained by the UCI Machine Learning database.

The "spam" concept is diverse: advertisements for products/web sites, make money fast schemes, chain letters, pornography…

Our collection of spam e-mails came from our postmaster and individuals who had filed spam. Our collection of non-spam e-mails came from filed work and personal e-mails, and hence the word 'george' and the area code '650' are indicators of non-spam. These are useful when constructing a personalized spam filter. One would either have to blind such non-spam indicators or get a very wide collection of non-spam to generate a general purpose spam filter.

**Attribute Information:** The last column of 'spambase.dat' denotes whether the e-mail was considered spam (1) or not (0), i.e. unsolicited commercial e-mail. Most of the attributes indicate whether a particular word or character was frequently occuring in the e-mail. The run-length attributes (55-57) measure the length of sequences of consecutive capital letters. For the statistical measures of each attribute, see the end of this file. Here are the definitions of the attributes:

48 continuous real [0,100] attributes of type word_freq_WORD = percentage of words in the e-mail that match WORD, i.e. 100 * (number of times the WORD appears in the e-mail) / total number of words in e-mail. A "word" in this case is any string of alphanumeric characters bounded by non-alphanumeric characters or end-of-string.

6 continuous real [0,100] attributes of type char_freq_CHAR] = percentage of characters in the e-mail that match CHAR, i.e. 100 * (number of CHAR occurences) / total characters in e-mail

1 continuous real [1,…] attribute of type capital_run_length_average = average length of uninterrupted sequences of capital letters

1 continuous integer [1,…] attribute of type capital_run_length_longest = length of longest uninterrupted sequence of capital letters

1 continuous integer [1,…] attribute of type capital_run_length_total = sum of length of uninterrupted sequences of capital letters = total number of capital letters in the e-mail

1 nominal {0,1} class attribute of type spam = denotes whether the e-mail was considered spam (1) or not (0), i.e. unsolicited commercial e-mail.

**Your task** Irrespective of the variables included, the task in this homework is to build a model for prediction which we want as precise as possible.

To this end:

1. Split randomly the data set in a train and a test set (take 1000 observations for the test set using `set.seed(1)`)
2. Using the training data, fit the following models:
    a. a pruned tree using the default settings of the function 'tree()'.
    b. a pruned tree obtained by adding the following `option` to the function `tree()`: `control=tree.control(nrow(spam.` `mincut = 2, minsize = 5, mindev = 0.001)`. This option allows to control the growth of the tree. To have more information about this function see the help of the library `tree`.
    c. a tree bagging model
    d. a random forest model
    e. a k-NN classification (use the test set to determine the optimal k)
3. Using the test data:
    a. Construct the ROC curve for each of the models determined in 2.
    b. Report all curves of the same plot
    c. Compare your models and discuss which one you would choose. For this example `k-NN` classification should perform much worse with repect to tree models. Can you try to explain why? (Hint: on the ISLR book or from some other source, look for the `curse of dimensionality`)

**Hints for part 3** In order to construct the ROC curves one needs to use the vector of predicted probabilities for the test data. The usage of the function `predict()` may be different from model to model.

h1. for trees the matrix of predicted probabilities (for Good and Spam) will be provided by using

```
predict(tree.model,test.data)
```

Put the output in a `data.table` to access it properly

h2. for Random Forests (and bagging) the matrix of predicted probabilities (for Good and Spam) will be provided by using

```
predict(tree.model,test.data,type="prob")
```

Again, put the output in a `data.table` to access it properly

h.3 for k-NN one needs to add the option `prob=TRUE` to the function `knn()` and use the function `attributes` to access them. An example is provided in the code below: the vector of SPAM probabilities are in the `knn.p` object. Here I used `k=7`; if the optimal `k` you obtain for your model is less than `7`, use anyway the value provided here to have a good representation of the ROC curve.

```
knn.pred=knn(x.train,x.test,y.train,k=7,prob=TRUE)
knn.p=1-attributes(knn.pred)$prob
```

## Additional exercises for PStat 231

### Question 2

Consider the Gini index, classification error, and cross-entropy in a simple classification setting with two classes. Create a single plot that displays each of these quantities as a function of $\hat{p}_{m1}$. The x-axis should display $\hat{p}_{m1}$, ranging from `0`to `1`, and the y-axis should display the value of the Gini index, classification error, and entropy.

Use R to produce your plot.

**Question 3**

Suppose we produce ten bootstrapped samples from a data set containing red and green classes. We then apply a classification tree to each bootstrapped sample and, for a specific value of $X$, produce 10 estimates of $P(\text{Class is Red}|X)$:

0.1, 0.15, 0.2, 0.2, 0.55, 0.6, 0.6, 0.65, 0.7, and 0.75.

There are two common ways to combine these results together into a single class prediction. One is the majority vote approach discussed in class (and chapter 8 of ISLR). The second approach is to classify based on the average probability. In this example, what is the final classification under each of these two approaches?