BUSINESS DATA MINING (IDS 572)

HOMEWORK 3
DUE DATE: WEDNESDAY, SEPTMEBER 21 AT 03:00 PM

- Please provide succinct answers to the questions below.
- You should submit an electronic pdf or word file in blackboard.
- Please include the names of all team-members in your write up and in the name of the file.
- Please include all R codes used to answer the questions.

In this assignment we use the spam data set in the ElemStatLearn library. Load the data set into memory with data(spam).

```
> install.packages("ElemStatLearn")
> library(ElemStatLearn)
> data(spam)
```

The data is for learning to classify e-mail as spam or real mail. There are 58 columns: 57 of them are features (see help(spam)), and the last one is a categorical variable ("factor"), called spam, with two values, email and spam. There are 4601 rows, representing 4601 different e-mails. You will also need the packages tree and adabag.

Problem 1. To see whether a classifier is actually working, we should compare it to a constant classifier which always predicts the same class, no matter what the input features actually are.

(a) What fraction of the e-mails are actually spam?
(b) What should the constant classifier predict?
(c) What is the error rate of the constant classifier?

Problem 2. Divide the data set at random into a training set of 2301 rows and a testing set of 2300 rows. Check that the two halves do not overlap (use intersect() function), and that they have the right number of rows. What fraction of each half is spam? (Do not hand in a list of 2301 row numbers.)

Problem 3. Remember to show your work by including your code.

(a) Fit a classification tree to the training data. Prune the tree by cross-validation (see below). Include a plot of the CV error versus tree size, a plot of the best tree, and its error rate on the testing data. Which variables appear in the tree?
(b) Use bagging to fit an ensemble of 100 trees to the training data. Report the error rate of the ensemble on the testing data. Include a plot of the importance of the variables, according to the ensemble.
(c) Which (if any) of these methods out-performs the constant classifier?

Problem 4. Pick the prediction method from the previous problem with the lowest error rate.

(a) What fraction of the spam e-mails in the training set did it not classify as spam?
(b) What fraction of the genuine e-mails in the testing set did it classify as spam?
(c) What fraction of e-mails it classified as spam were actually spam?

****

R advice

The tree package contains functions prune.tree and cv.tree for pruning trees by cross-validation.
The function prune.tree takes a tree you fit by tree, and evaluates the error of the tree and various prunings of the tree, all the way down to the stump (1-rules). The evaluation can be done either on new data, if supplied, or on the training data (the default). If you ask it for a particular size of tree, it gives you the best pruning of that size. If you don't ask it for the best tree, it gives an object which shows the number of leaves in the pruned trees, and the error of each one. This object can be plotted.

```
my.tree = tree(y ~ x1+x2, data=my.data) # Fits tree
prune.tree(my.tree, best=5) # Returns best pruned tree with 5 leaves, evaluating error on training data
prune.tree(my.tree, best=5, newdata=test.set) # Ditto, but evaluates on test.set
my.tree.seq = prune.tree(my.tree) # Sequence of pruned tree sizes/errors
plot(my.tree.seq) # Plots size vs. error
my.tree.seq$dev # Vector of error rates for prunings, in order
my.tree.seq$size[which.min(my.tree.seq$dev)] # Size of best tree
```

Finally, prune.tree has an optional method argument. The default is method="deviance", which fits by minimizing the negative log likelihood. You may get better results by saying method="misclass", which looks at the misclassification rate. The function cv.tree does k-fold cross-validation (default is 10). It requires as an argument a fitted tree, and a function which will take that tree and new data. By default, this function is prune.tree.

```
my.tree.cv = cv.tree(my.tree)
```

The type of output of cv.tree is the same as the function it's called on. Optional arguments to cv.tree can include K, and any additional arguments for the function it applies.

The package adabag includes routines for bagging and boosting of trees. (It may require you to install some other packages it depends on as well.) Bagging is done with the command bagging:

```
my.bag = bagging(y ~ x1+x2, data=my.data)
```

This returns an object of the class bagging, which has a prediction method. Note that the prediction method doesn't just return the predictions, but also some error evaluation see help(predict.bagging). Optional arguments to bagging include mfinal (the number of trees to include in the ensemble, default

100), minsplit (minimum number of observations per node, default 5), cp (minimum amount the fit has to increase by to make a split, default 0.01), and maxdepth (maximum depth of any one tree, default 2 for binary data). You will want to play with these defaults to get a good fit.

The output of bagging is an object which contains all the trees in the ensemble, the predicted value of the class, and the relative importance of each input variable, measured by the number of internal nodes which split on it.