Data Mining 2014
Assignment 1: Classification Trees

Instructions

This assignment should be made in teams of two students. Solutions should be handed in ultimately on Friday October 10th. Send your solution by e-mail to `A.J.Feelders@uu.nl`. Your solution consists of

1. An R workspace containing your program code and data set.
2. A flat ascii file containing the documented program code, and
3. A PDF file containing a short report (2 pages) of your analysis.

Always put name and student number on your work.

Assignment

Write a function in R to grow a classification tree. Also write a function that uses this tree to predict the class label for given attribute values. More specifically you should write two main functions, with the names `tree.grow` and `tree.classify`. The function `tree.grow` has input arguments `x, y, nmin, minleaf`. Here `x` is a data matrix containing the attribute values. Each row of `x` contains the attribute values of one training example. You may assume that all attributes are numeric, or binary with values coded as `0` and `1`. `y` is the vector of class labels. You may assume that the class label is binary, with values coded as `0` and `1`. Furthermore, you may assume there are no missing values (either in training or prediction).

The parameters `nmin` and `minleaf` (both integers) are used to stop growing the tree early, to prevent over-fitting (we do not consider *pruning* in this assignment). `nmin` is the number of observations that a node must contain at least, for it to be allowed to be split (that is, if it contains fewer cases than `nmin`, it becomes a leaf node). `minleaf` is the minimum number of observations required for a leaf node; hence a split that creates a node with fewer than `minleaf` observations is not acceptable. If the algorithm performs a split, it should be the best split that meets the `minleaf` constraint. Use the gini index for determining the quality of a split.

The function `tree.grow` should return a "tree object" that can be used for predicting new cases. You are free to choose the data structure as long as it can be used for predicting new cases in the following way. A new case is dropped down the tree, and assigned to the majority class of the leaf node it ends up in. More precisely, the function `tree.classify` has input arguments `x` and `tr`. Here `x` is a data matrix containing the attribute values of the cases for which predictions are required, and `tr` is a tree object created with the function `tree.grow`. The function `tree.classify` has a single output argument `y`, which is the vector of predicted class labels for the cases in `x`.

Finally, apply your algorithm to a data set of your choice (e.g. from the UCI Machine Learning repository), but not the Pima indians data. Make sure that the data set you choose has the right properties to be analyzed by your algorithm. Make a training and test sample, construct your model on the training sample, and estimate its error rate on the test sample. Use about 70% of the data for training and the remaining 30% for testing. The training set should be a *random* sample from the data set. Try different settings of `nmin` and `minleaf` and observe how it affects the error rate. Try to find the best settings. Describe your analysis of the chosen data set in a 2 page report that includes a description of the data set, and a table with the different settings of `nmin` and `minleaf` you tried and the corresponding error rate on the test set.

Handing in the assignment

The R workspace (file with extension `.Rdata`) that you hand in should be tested to work on the Windows machines in the computer labs of the University. This file will be used to test the functions you have written.

Also put the functions you have written in a flat ascii file. Before you give the code of a function, provide the following information (start each line containing this information with the symbol #):

1. Name of the function.
2. Names and types of input arguments.
3. The result returned by the function.
4. A short description of what it does.

The main functions should be called `tree.grow` and `tree.classify`, and should be the first two functions in the file you submit. Any other functions you have written that are needed to get things working should be listed below that. Your report on the analysis of the chosen data set should be handed in in PDF format.

Grading

The following considerations are taken into account to determine the grade for this assignment:

- Does the program work, and does it return the correct result?
- Efficiency and elegance of the implementation.
- Quality of the report.

Some Hints

- We warmy recommend that you work with Rstudio, an integrated development environment for R.
- Read "Getting started with R" on the course web page first, and make the practice assignments. Play around a little bit with R before you start with the "real work".
- During tree construction, a node in the tree is in fact "nothing more" than a subset of the training examples. Such a subset can be represented by a vector of *row numbers* of the observations contained in the subset.
- If you want help on a particular topic, type `help(topic)` on the `R` command line. For example,

  ```
  > help(sort)
  ```

  gives you information on the `sort` function.
- To test your algorithm you could apply it to the credit scoring data set used in the lectures. With `nmin = 2` and `minleaf = 1` you should get the same tree as presented in the lecture slides.
- For a more elaborate test, use the Pima indians data from the UCI machine learning repository. If you grow the tree on the complete data set with `nmin = 20` and `minleaf = 5`, and you use this tree to predict the training sample itself, you should get the following confusion matrix (row: true class, column: predicted class):

	0	1
0	444	56
1	54	214

  If the confusion matrix produced by your algorithm differs substantially from this one, there is probably an error in your code.