FIN 510 Big Data Analytics in Finance

Lab 13: Classification Trees

Due on 10/16/2021

Identifying competitive auctions on eBay.com

The file eByAuctions.csv contains information on 1972 auctions transacted on eBay.com during May-June 2004. The goal is to use this data to build a model that will distinguish competitive auctions from non-competitive auctions. A competitive auction is defined as an auction with at least two bids placed on the item being auctioned. The binary outcome variable (Competitive.) takes on a value of 1 if the auction is competitive and a value of 0 if it is not competitive. The data includes variables that describe the item (e.g., auction category), the seller (e.g., his or her eBay rating), and the auction terms that the seller selected (e.g., auction duration, opening price, currency, and day of week that the auction closed). In addition, we include the price at which the auction closed. The following table describes each of the predictors and the response.

DESCRIPTION OF VARIABLES FOR AUCTIONS ON EBAY.COM EXAMPLE	
Competitive.	Whether the auction had a single bid (0) or more (1)
Category	Category of the auctioned item
currency	Currency
sellerRating	A rating by eBay, as a function of the number of good and bad transactions
Duration	Number of days the auction lasted
endDay	Day of week that the auction closed
ClosePrice	Price item sold
OpenPrice	Initial price set by the seller

0) Load the packages

Use library() to load rpart, rpart.plot, and caret.

1) Create a data frame

Load the data with read.csv(). Use parameter stringsAsFactors = TRUE to convert character variables to categorical variables. Save the result in a data frame named ebay.df.

Return the first six rows and column names using head() and names().

2) Convert numeric variables to categorical variables

Auction duration (Duration) takes on 5 numeric values: 1, 3, 5, 7, and 10. The outcome variable (Competitive.) takes on 2 numeric values: 0 and 1. Convert Duration and Competitive. to categorical or factor variables using as.factor().

Use str() to return the structure of the data frame. Duration should be a factor variable with 5 levels, and Competitive. should be a factor variable with 2 levels.

3) Data partition

Partition the data into training (60%) and test (40%) sets: use set.seed(1) to set the random seed and sample() to take a sample of row numbers for the training set. Save a sample of row numbers, the training set, and the testing set as train.index, train.df and test.df, respectively.

Hint: dim(ebay.df)[1] returns the total number of the rows in the data frame, 0.6* dim(ebay.df)[1] specifies the number of rows to select for the training set, and c(1:dim(ebay.df)[1]) represents row numbers.

4) Fit a classification tree on the training set

Use set.seed(1) to set the random seed such that the result in question 7 can be reproduced.

To predict whether an auction is competitive or not, fit a classification tree with all predictors using rpart() with method="class". To avoid overfitting, set the minimum number of records in a terminal node to 30 (minbucket=30). Also, set the smallest value of the complexity parameter to 0.00001 (cp=0.00001). Save the classification tree as ct.

5) Generate predicted classes for records in the test set

Use predict() with type = "class“ to compute predicted classes for auctions in the test set.

Save the predicted classes as pred.class and return the first six values using head().

6) Create a confusion matrix for the test set

Use confusionMatrix() to create a confusion matrix for the test set.

The first parameter in the function is a factor of predicted classes and the second parameter is a factor of actual classes. Notice that both predicted classes and actual classes are already factors.

Given that the outcome variable Competitive. is "1" for competitive auctions, set the third parameter named positive to "1" to specify the positive class.

7) Prune the tree

Use ct$cptable to display various complexity parameter values and their cross-validated errors. Since we did not specify the value of the xval parameter in question 4, it uses 10-fold cross-validation by default.

Return the cp value that corresponds to the lowest cross-validated error (xerror).

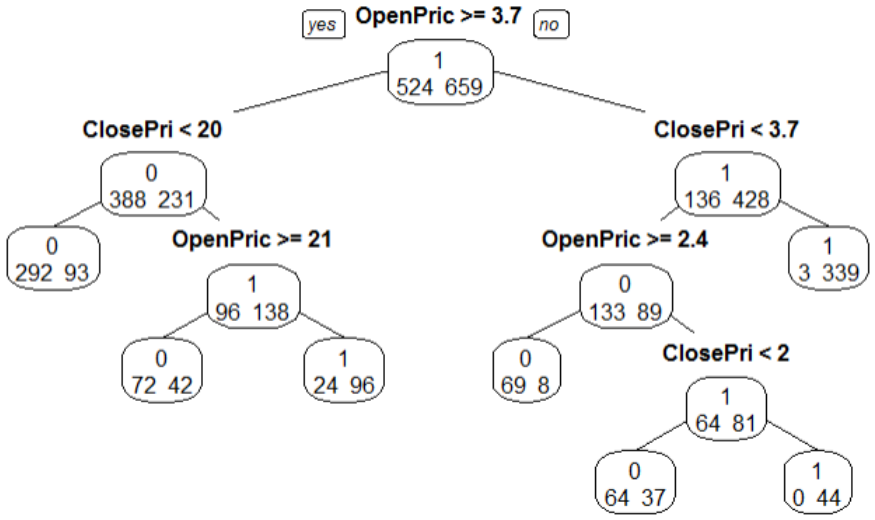
Hint: which.min(ct$cptable[,"xerror"]) returns the index of the row that contains the minimum cross-validated error. ct$cptable[which.min(ct$cptable[,"xerror"]),"CP"] subsets the cp table by the row index and the column name.

8) Identify the best-pruned tree

Find the pruned tree using prune(). The first parameter in the function is ct and the second parameter is the cp value that yields the lowest cross-validated error computed in question 7. Save the pruned tree as pruned.ct.

9) Plot the pruned tree

Plot the pruned tree using prp(). Set type to 1 to label all nodes and set extra to 1 to display the number of observations that fall in the node.



10) Generate predicted classes for records in the test set

According to the pruned tree, use predict() with type = "class“ to compute predicted classes for auctions in the test set.

Save the predicted classes as pruned.pred.class and return the first six values using head().

11) Create a confusion matrix for records in the test set

According to the pruned tree, use confusionMatrix() to create a confusion matrix in the test set.

The first parameter in the function is a factor of predicted classes and the second parameter is a factor of actual classes. Notice that both predicted classes and actual classes are already factors.

Given that the outcome variable Competitive. is "1" for competitive auctions, set the third parameter positive to "1" to specify the positive class.