FIN 510 Big Data Analytics in Finance

Lab 15: Bagging, Random Forests, and Boosting

Due on 10/23/2021

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 adabag, randomForest, caret, and gains.

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 length 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) Bagging

4.1) Fit a bagging algorithm on the training set

Use set.seed(1) to set the random seed.

To predict whether an auction is competitive or not, fit a bagging algorithm with all predictors using bagging(). Save the bagged tree as bag.

4.2) Generate predicted probabilities and classes for records in the test set

According to the bagging algorithm, use predict() to make predictions for auctions in the test set. Save the resulting object as bag.pred.

Return the first six values of predicted probabilities and classes using bag.pred$prob and bag.pred$class, respectively.

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

Use confusionMatrix() to create a confusion matrix for records 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 should be factors. Use as.factor() to convert predicted classes from character values to categorical values.

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

5) Random Forests

5.1) Fit a random forests algorithm on the training set

Use set.seed(1) to set the random seed.

To predict whether an auction is competitive or not, fit a random forests algorithm with argument mtry=4 using randomForest(). Therefore, a random sample of 4 predictors are chosen from the full set of predictors as candidates at each split. Save the random forests as rf.

5.2) Variable importance plot

Plot the importance of the variables using varImpPlot(rf).

5.3) Generate predicted probabilities and classes for records in the test set

According to the random forests algorithm, use predict() with type = "prob" to compute predicted probabilities for auctions in the test set. Save the resulting matrix as rf.pred.prob and return the first six rows using head(). Notice that rf.pred.prob[,2] represents the predicted probabilities of being competitive auctions.

Use predict() with type = "class“ to compute predicted classes for auctions in the test set. Save the resulting matrix as rf.pred.class and return the first six values using head().

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

Use confusionMatrix() to create a confusion matrix for records 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 named positive to "1" to specify the positive class.

5.5) Create a gain table

Use gains() from the gain library to rank records in the test set based on the predicted probabilities of being competitive and group auctions into 10 groups. Save the result as gain.

Hint: the first parameter in the function gains() is a vector of actual classes (=1 for competitive auctions and 0 for non-competitive auctions). The second parameter is a vector of predicted probabilities of being competitive auctions, which is computed in question 5.3. Set the third parameter groups to 10.

Notice that the vector of actual classes should be numeric. Use as.numeric(as.character()) to convert categorical values to numeric values.

Try the following values returned from the object gain:

gain$cume.pct.of.total returns the cumulative percentage of competitive auctions.

gain$cume.obs returns the cumulative number of auctions.

5.6) Plot a lift chart

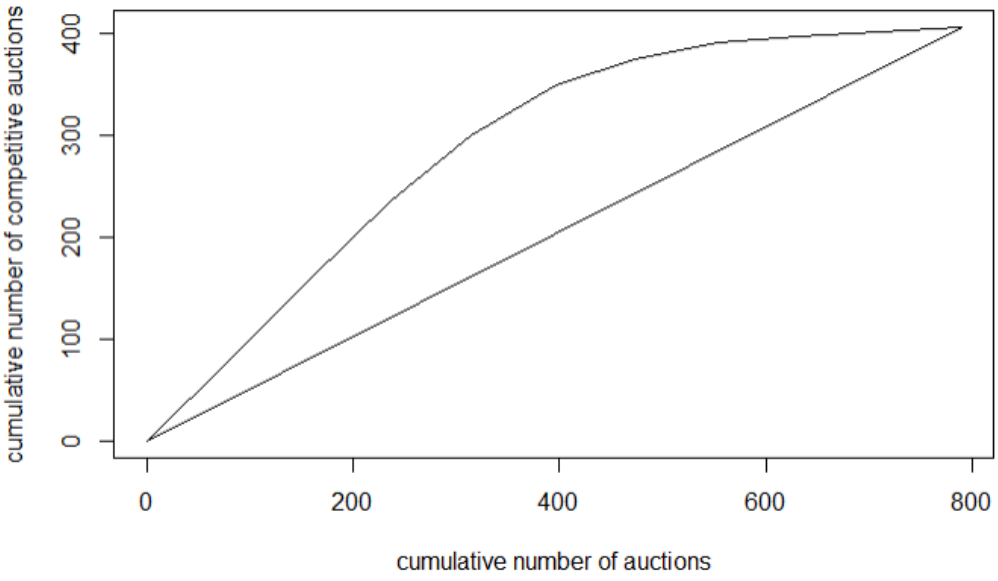
Use plot() to plot the cumulative number of competitive auctions against the cumulative number of auctions.

Hint: gain$cume.pct.of.total*sum(test.df$Competitive.==1) returns the cumulative number of competitive auctions, where sum(test.df$Competitive.==1) represents the total number of competitive auctions. Therefore, c(0,gain$cume.pct.of.total*sum(test.df$Competitive==1)) are the y-axis values.

Specify the x-axis label as “cumulative number of auctions", y-axis label as "cumulative number of competitive auctions", and the type is a line plot (type="l").

Use lines() to add a diagonal benchmark line, which represents a naïve prediction for each auction and accumulates the average value of competitive auctions in each group.

Hint: dim(test.df)[1] returns the total number of auctions. Therefore, c(0, dim(test.df)[1]) are the x-axis values.



6) AdaBoost

6.1) Fit an adaptive boosting algorithm on the training set

Use set.seed(1) to set the random seed.

To predict whether an auction is competitive or not, fit a AdaBoost algorithm with all predictors using boosting(). Save the boosted tree as boost.

6.2) Generate predicted probabilities and classes for records in the test set

According to the AdaBoost algorithm, use predict() to make predictions for auctions in the test set. Save the resulting object as boost.pred.

Return the predicted probabilities and class using boost.pred$prob and boost.pred$class, respectively.

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

Use confusionMatrix() to create a confusion matrix for records 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 should be factors. Use as.factor() to convert predicted classes from character values to categorical values.

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