FIN 510 Big Data Analytics in Finance

Lab 8: Logistic Regression

Due on 09/25/2021

Identifying competitive auctions on eBay.com

The file eBayAuctions.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 and suppress scientific notation

Use library() to load caret and gains. Use options(scipen=999) to suppress scientific notation.

1) Create a data frame

Load the data with read.csv(). Use parameter stringsAsFactors = FALSE to prevent character variables from converting to categorical variables, because we will modify the values of character variables later. Save the result in a data frame named ebay.df.

Use head() and names() to return the first six rows and column names.

2) Apply a function to each group

Compute the mean of the binary outcome variable (ebay.df$Competitive.) for each day of the week that the auction closed (ebay.df$endDay).

Hint: function tapply() applies a function to each group. The first parameter in the function is the binary outcome variable, the second parameter is the variable used for grouping, and the third parameter is the function (mean).

Compute the mean of the binary outcome variable (ebay.df$Competitive.) for each auction category (ebay.df$Category).

3) Reduce levels in the day of week that the auction closed (endDay)

To reduce the number of dummies that will be used in the model, categories that appear most similar with respect to the distribution of competitive auctions (Competitive.) could be combined.

The auction ending days "Sun" and "Fri" have approximately equal average values of Competitive. Therefore, combine them into a single category called "Sun_Fri".

Hint: ebay.df$endDay == "Sun" selects rows where the value in the variable endDay is “Sun”. ebay.df$endDay[ebay.df$endDay == "Sun"] <- "Sun_Fri" assigns a new value to endDay for these rows.

4) Reduce levels in auction categories (Category)

The categories "Business/Industrial" and "Computer" have the same average values of the binary variable "Competitive.". Hence, we can combine the two categories into a single category called "Computer".

Similarly, the average values of "Competitive." for categories "Antique/Art/Craft" and "Collectibles" are almost the same. Thus, we can combine them into a single category called "Collectibles".

Hint: assign a new value "Computer" to the variable Category for rows where the Category value is "Business/Industrial". Assign a new value " Collectibles " to the variable Category for rows where the Category value is "Antique/Art/Craft ".

5) Convert Duration to a categorical or factor variable

Auction duration (Duration) takes on 5 numeric values: 1, 3, 5, 7, and 10. Convert Duration to a categorical or factor variable using as.factor().

Use str() to return the data type of variables in the data frame. Duration should be a factor variable with 5 levels. Logistic regression models will automatically create four dummy variables from the factor's five levels.

6) 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 tran.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.

7) Fit a logistic regression model

To predict whether an auction is competitive or not, fit a logistic regression model with all predictors using glm() with family = "binomial". Save the result as reg.

Use summary() to print the summary table of the regression.

Notice that 4 dummies variables are created for the categorical variable Duration, which has 5 levels. Dummy variables are also automatically created for character variables such as Category, currency, and endDay.

8) Generate predicted probabilities

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

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

9) Create a confusion matrix

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

Use 0.5 as the threshold or cutoff value: an auction is competitive if the predicted probability is more than 0.5 and an auction is not competitive if the predicted probability is less than or equal to 0.5.

Hint: ifelse(pred > 0.5, 1, 0) returns 1 if the predicted probability is more than 0.5 and 0, otherwise.

The first parameter in function confusionMatrix() from the caret library is a factor of predicted classes and the second parameter is a factor of actual classes. Use as.factor() to convert observed and predicted classes to 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.

10) 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 computed in question 8. Set the third parameter named groups to 10.

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.

11) 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.) returns the cumulative number of competitive auctions, where sum(test.df$Competitive.) represents the total number of competitive auctions. Therefore, c(0,gain$cume.pct.of.total*sum(test.df$Competitive.)) 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.