FIN 550 Big Data Analytics in Finance

Lab 9: Discriminant Analysis

Due on 02/24/2024

Identifying good system administrators

A management consultant is studying the roles played by experience and training in a system administrator's ability to complete a set of tasks in a specified amount of time. In particular, she is interested in discriminating between administrators who are able to complete given tasks within a specified time and those who are not. The fille SystemAdministrators.csv contains information regarding the performance of 75 randomly selected administrators.

The goal is to perform a discriminant analysis to identify good system administrators (Completed.task=Yes) using two predictors: Experience and Training. The binary outcome variable (Completed.task) is either Yes or No, according to whether or not the administrator completed the tasks. The predictor Experience measures months of full-time system administrator experience, while the predictor Training measures the number of relevant training credits.

0) Load the packages

Use library() to load DiscriMiner, caret, and gains.

1) Create a data frame

Load the data with read.csv(). Save the result in a data frame named df.

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

2) 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 of the training set. Save the row numbers of the training set as train.index.

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

Use setdiff() to return the row numbers of the test set and save the result as test.index.

3) Perform a discriminant analysis using the training set

To classify system administrators into good administrators (Completed.task=Yes) and poor administrators (Completed.task=No), use linDA() to run a discriminant analysis with both predictors. Save the result as da.reg.

Hint: df[train.index,1:2] represents the predictors in the training set and df[train.index,3] represents the outcome variable in the training set.

Display the estimated classification functions, classification scores for poor administrators, classification scores for good administrators, and predicted classes for records in the training set.

4) Make predictions for records in the test set

Use classify() to classify observations in the test set based on discriminant analysis object. Save the result as pred.

Hint: df[test.index,1:2] represents the predictors in the test set.

Display the classification scores and predicted classes for records in the test set. Using the classification scores, calculate the predicted probabilities of being a good administrator and save the result as prob.good.

5) Create a confusion matrix using the test set

Use function confusionMatrix() to create a confusion matrix. The first parameter in the function is a factor of predicted classes and the second parameter is a factor of actual classes. Given that the outcome variable Completed.task is "Yes" for good administrators, set the third parameter positive to "Yes" to specify the positive class.

Hint: pred$pred_class and df[test.index,3] represent the predicted and actual classes for records in the test set. Notice that predicted classes are already factors. Use as.factor() to convert actual classes to factors.

6) Create a gain table

Use gains() from the gain library to rank observations in the test set based on the predicted probabilities of being good administrators and group administrators into 5 groups. Save the result as gain.

Hint: The first parameter in the function gains() represents a vector of actual classes and it should be a vector of numeric values. Since the values in the outcome variable (Completed.task) is a character variable, use ifelse(df[test.index,3]=="Yes",1,0) as the first parameter to assign 1 to Yes and 0 to No. The second parameter is a vector of predicted probabilities of being good administrators computed in question 4. Set the third parameter named groups to 5.

Try the following values returned from the object gain:

gain$cume.pct.of.total returns the cumulative percentage of good administrators.

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

7) Plot a lift chart

Use plot() to plot the cumulative number of good administrators against the cumulative number of administrators.

Hint: gain$cume.pct.of.total*sum(df[test.index,3]=="Yes") returns the cumulative number of good administrators, where sum(df[test.index,3]=="Yes") represents the total number of good administrators. Therefore, c(0,gain$cume.pct.of.total*sum(df[test.index,3]=="Yes")) are the y-axis values.

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

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

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