FIN 510 Big Data Analytics in Finance

Lab 16: Neural Nets

Due on 10/23/2021

Direct mailing to airline customers

East-West Airlines has entered into a partnership with the wireless phone company Telcon to sell the latter's service via direct mail. The goal is to run a neural net model to classify East-West customers as to whether they purchase a wireless phone service contract.

The file EastWestAirlinesNN.csv contains a subset of a data sample of who has already received a test offer. The binary outcome variable (Phone_Sale) takes on a value of 1 if a customer purchased Telcon service as a result of the direct mail campaign and a value of 0 if a customer did not make a purchase. The following table describes each of the predictors.

DESCRIPTION OF VARIABLES FOR DIRECT MAILING TO AIRLINE CUSTOMERS EXAMPLE	
TopFlight	Topflight status (1=yes, 0=no)
Balance	Number of miles eligible for award travel
Qual_miles	Number of miles counted as qualifying for Topflight status
cc1_miles.	Number of miles earned with freq. flyer credit card in the past 12 months (1=under 5,000; 2=5,000–10,000; 3=10,0001-25,000; 4=25,001-50,000; 5=over 50,000)
cc2_miles.	Number of miles earned with Rewards credit card in the past 12 months (same mile bins as cc1_miles)
cc3_miles.	Number of miles earned with Small Business credit card in the past months (same mile bins as cc1_miles)
Bonus_miles	Number of miles earned from non-flight bonus transactions in the in the past 12 months
Bonus_trans	Number of non-flight bonus transactions in the past 12 months
Flight_miles_2mo	Number of flight miles in the past 12 months
Flight_trans_12	Number of flight transactions in the past 12 months
online_12	Number of online transactions in the past 12 months
Email	E-mail address on file (1= yes, 0 =no)
Club_member	Member of the airline's club (1=yes, 0=no)
Any_cc_miles_12mo	Dummy variable indicating whether member added miles on any credit card type within the past 12 months (1=yes, 0=no)

0) Load the packages

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

1) Create a data frame

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

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

2) Create two outcome dummies to represent output nodes

The binary outcome variable (Phone_Sale) takes on a value of 1 if a customer purchased a phone service contact via direct mail and a value of 0 if a customer did not make a purchase.

Create a dummy variable named purchase in df which is TRUE if Phone_Sale is equal to 1 and FALSE, otherwise. Create another dummy variable named not_purchase in df which is TRUE if Phone_Sale is equal to 0 and FALSE, otherwise.

Hint: df$Phone_sale==1 returns TRUE if Phone_Sale is equal to 1 and FALSE, otherwise. df$Phone_sale==0 returns TRUE if Phone_Sale is equal to 0 and FALSE, otherwise.

3) Remove unnecessary variables

Drop the first column that contains information about customers' ID. Save the updated dataframe as df. Return column names using names().

Hint: df[, -1] subset the data frame by excluding the first column.

4) Remove rows with missing values

Use na.omit() to remove rows with any missing values. Save the updated dataframe as df.

Hint: na.omit(df) returns the data frame df with incomplete observations removed.

5) 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(df)[1] returns the length of the 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.

6) Normalize predictors to a scale of [0, 1]

Notice that some predictors are not on a scale of [0,1]. Before fitting a neural network to data, we need to scale predictors to a [0, 1] interval.

First, use preProcess() with method="range" to estimate the transformation. The estimate should be based on the predictors in the training set. Save the object as norm.values.

Hint: train.df[,c(1:14)] represents all of the predictors in the training set.

Second, use function predict() to normalize the predictors in the training set and save the result as train.norm.df.

Last, use function predict() to normalize the predictors in the test set and save the result as test.norm.df.

Hint: test.df[,c(1:14)] represents all of the predictors in the test set.

7) Fit a neural network model on the training set

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

To classify East-West customers as to whether they purchase a wireless phone service contract, we will fit a neural network with a single hidden layer with 5 nodes on the training set using neuralnet().

Specify output nodes as train.df$not_purchase and train.df$purchase in order.

Specify the following input nodes in order: Topflight, Balance, Qual_miles, cc1_miles., cc2_miles., cc3_miles., Bonus_miles, Bonus_trans, Flight_miles_12mo, Flight_trans_12, Online_12, Email, Club_member, and Any_cc_miles_12mo. Use the normalized predictors by specifying data= train.norm.df.

Given that this is a classification problem, set parameter linear.output to FALSE. Set parameter hidden to 5 to specify the number of nodes in the single hidden layer. Save the result as nn.

Use plot(nn, rep = "best") to plot the network and nn$weights to display the weights.

8) Generate predicted probabilities for records in the test set

According to the neural network, use compute() and normalized predictors (test.norm.df) to make predictions for customers in the test set. Save the resulting object as nn.pred.

Return the predicted probabilities of making purchases using nn.pred$net.result[,2] and save the result as nn.pred.prob.

Hint: Given that we have specified output nodes in order in question 7, nn.pred$net.result has two columns: the first column represents predicted probabilities of not making purchases and the second column represents predicted probabilities of making purchases.

9) Generate predicted classes for records in the test set

According to the predicted probabilities of making purchases, generate the predicted classes (1 for making purchases and 0 for not making purchases) for customers in the test set using a cutoff value of 0.5. Save the result as nn.pred.class.

Hint: ifelse(nn.pred.prob>0.5,1,0) returns 1 if the probability of making purchases is more than 0.5, and 0, otherwise.

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

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

The first parameter in the function confusionMatrix() from the caret library is a factor of predicted classes calculated in question 9 (nn.pred.class). The second parameter is a factor of actual classes in the test set (test.df$Phone_sale). Notice that both predicted classes and actual classes should be factors. Use as.factor() to convert predicted classes and actual classes from numeric values to categorical values.

Given that the outcome variable Phone_sale is "1" for customers who make purchases, set the third parameter positive to "1" to specify the positive class.

11) Create a gain table

Use gains() from the gain library to rank customers in the test set based on the predicted probabilities of making purchases, and group customers into 10 groups. Save the result as gain.

Hint: the first parameter in function gains() is a vector of actual classes (test.df$Phone_sale). The second parameter is a vector of predicted probabilities computed in question 8 (nn.pred.prob). 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 customers who make purchases.

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

12) Plot a lift chart

Use plot() to plot the cumulative number of customers who make purchases against the cumulative number of customers.

Hint: gain$cume.pct.of.total*sum(test.df$Phone_sale) returns the cumulative number of customers who make purchases, where sum(test.df$Phone_sale) represents the total number of customers who make purchases. Thus, c(0,gain$cume.pct.of.total*sum(test.df$Phone_sale)) are the y-axis values.

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

Use lines() to add a diagonal benchmark line, which represents a naïve prediction for each customer and accumulates the average value of customers who make purchases in each group. Hint: dim(test.df)[1] returns the total number of customers. Therefore, c(0, dim(test.df)[1]) are the x-axis values.