Training Multinomial Regression Model in R

Machine Learning and Applications

Tufan Bostan

26 03 2021

In this work, multinomial regression model will be trained using R. Data to be used here is "HR" from "DALEX" package. Structure of the dataset is based on a real data, from Human Resources department with information which employees were promoted, which were fired. Based on the information's of the employees, an attempt will be made to decision whether they will be fired, not fired or promoted. First, some descriptive statistics will be obtained and interpreted. Then, the data will be split into two parts which are "Train" and "Test". Then using the "Train" part of data the MLRM methods will be applied. Finally, performance of the models on train and test sets will be interpreted.

Data Structure

First of all, "DALEX" package installed to access "HR" data (`install.packages("DALEX")`). After that the "DALEX" package must be called to access all of features of it. (`library(DALEX)`) . Now "HR" data introduced to R. The "HR" data assigned to "data". There were two variables that were not correctly classified. Before the model training process, these variables types changed numeric to factor. First 6 observations of the data are listed below. Then, structure of the data displayed. (`str(data)`) After that, data summary displayed to gather more information about variables. (`summary(data)`)

```
#Obtaining Data form its source:
#install.packages("DALEX")
library(DALEX)

data <- DALEX::HR                            #The data assigned to "data".
data$evaluation <-as.factor(data$evaluation) #Variable type chanced to factor
data$salary <-as.factor(data$salary)         #Variable type chanced to factor
head(data)                                   #First 6 row were obtained.

##   gender      age    hours evaluation salary   status
## 1   male 32.58267 41.88626          3      1    fired
## 2 female 41.21104 36.34339          2      5    fired
## 3   male 37.70516 36.81718          3      0    fired
## 4 female 30.06051 38.96032          3      2    fired
## 5   male 21.10283 62.15464          5      3 promoted
## 6   male 40.11812 69.53973          2      0    fired

#Structure of The Data:
str(data)

## 'data.frame':    7847 obs. of  6 variables:
##  $ gender    : Factor w/ 2 levels "female","male": 2 1 2 1 2 2 1 2 1 1 ...
##  $ age       : num  32.6 41.2 37.7 30.1 21.1 ...
##  $ hours     : num  41.9 36.3 36.8 39 62.2 ...
##  $ evaluation: Factor w/ 4 levels "2","3","4","5": 2 1 2 2 4 1 3 1 1 3 ...
##  $ salary    : Factor w/ 6 levels "0","1","2","3",..: 2 6 1 3 4 1 1 5 5 5 ...
##  $ status    : Factor w/ 3 levels "fired","ok","promoted": 1 1 1 1 3 1 3 2 1 3 ...

summary(data)                                 #To see levels of target variable and more
                                               information of others.

##     gender          age             hours        evaluation  salary
##  female:3949   Min.   :20.00   Min.   :35.00   2:2371     0:1105
##  male  :3898   1st Qu.:30.03   1st Qu.:37.64   3:2272     1:1417
```

```
##                Median :40.16   Median :46.28   4:1661     2:1461
##                Mean   :40.00   Mean   :49.71   5:1543     3:1508
##                3rd Qu.:49.96   3rd Qu.:59.48              4:1316
##                Max.   :60.00   Max.   :79.98              5:1040
##       status
##  fired   :2855
##  ok      :2221
##  promoted:2771
```

The data has 6 variables and 7847 observations. There are 4 factors and 2 numeric variables. Here, the target variable will be "status". The other variables are the independent variables, to be used to predict the "status". "gender" is representing gender of an employee and it's a factor, has 2 levels which are 3949 "female" and 3898 "male". "age" is representing age of an employee in the moment of evaluation, it has 40 mean, and ranges 20 to 60. "hours" is representing average number of working hours per week, it has 49,71 mean, and ranges 35 to 79.98. "evaluation" is representing evaluation in the scale 2(bad)-5(very good) and it's a factor, has 4 levels which are "2", "3", "4" and "5". they were observed 2371, 2272, 1661 and 1543 times, respectively. "salary" is representing level of salary in scale 0(lowest)-5(highest) and it's a factor, has 6 levels which are "0", "1", "2", "3", "4" and "5". They were observed 1105, 1417, 1461, 1508, 1316 and 1040 times, respectively. Finally, "status" is representing target variable, either "fired", "promoted" or "ok" (not fired and not promoted) and it's also a factor, has 3 levels which are "fired", "ok" and "promoted". They were observed 2822, 2221 and 2771 times, respectively. Here, it is possible to say that the number of observations at the "fired" and "promoted" levels of the "status" variable, are close to each other. However, since "ok" has fewer observations than the other levels, the possibility of encountering imbalance problem may occur.

Splitting Data

A sample was created using "HR" data. This sample has two parts one of them is going to be used for training to model and the other part for testing to the model. These are assigned to variables named `train` and `test`, respectively. Also, using `table(train$status);table(test$status)`, how many observations they contain is shown. `train` has 2261 "fired", 1812 "ok" and 2204 "promoted" observations and `test` 594 "fired", 409 "ok" and 567 "promoted" observations. After this process, the MLRM is ready to be generated.

```
#Splitting Data
set.seed(2380)                               #to get same sample every time
index <- sample(nrow(data),nrow(data)*0.8)   #Splitting Data
train <- data[index,]                        #assigning observations for train
test <- data[-index,]                        #assigning observations for test
table(train$status);table(test$status)       #displaying variable sizes

## 
##    fired       ok promoted 
##     2261     1812     2204

## 
##    fired       ok promoted 
##      594      409      567
```

As can be seen above, the data is divided into two parts. Still "ok" has fewer observations than the other levels. This has been mentioned before for main data. Although, the data is split, there is nothing changed. It may be still suspected that there is an imbalance problem.

Training MLRM

Before training the MLRM, the reference level of target variable must be determined. Here the reference level determined as “ok”. (`data$status <- relevel(data$status, ref = "ok")`). Then to use `multinom()` function, “nnet” is called by `library()` function. This function is essential for generating a MLRM. After all, the model is generated by `multinom()` function and the model summary were displayed.

```
#Training a MLRM
train$status <- relevel(train$status, ref = "ok")      #Reference level were determined.
library(nnet)                                          #To use "multinom()" function.
model <- multinom(status ~., data = train, trace = F)  #Model generated by train data
summary(model)

## Call:
## multinom(formula = status ~ ., data = train, trace = F)
##
## Coefficients:
##          (Intercept)  gendermale          age       hours evaluation3
## fired       5.320672 -0.05591252 -0.001405403 -0.08051062  0.03309601
## promoted   -7.597237 -0.00553580  0.001476226  0.11277513 -0.02549142
##          evaluation4 evaluation5    salary1     salary2    salary3    salary4
## fired      0.1079072   0.1335753 -1.5686349 -2.51859942 -2.4033252 -1.7620065
## promoted   3.5748057   3.4172130  0.2105901  0.05260612  0.1259842  0.1039802
##             salary5
## fired    0.07762555
## promoted 0.11155532
##
## Std. Errors:
##          (Intercept) gendermale         age       hours evaluation3 evaluation4
## fired      0.2677660 0.07153421 0.003092470 0.004072535  0.08251596   0.1200319
## promoted   0.3498403 0.08340161 0.003606793 0.004150432  0.11574080   0.1392968
##          evaluation5   salary1   salary2   salary3   salary4   salary5
## fired      0.1191669 0.1332500 0.1368514 0.1359633 0.1348701 0.1526782
## promoted   0.1377247 0.1685406 0.1657404 0.1653900 0.1706332 0.2028877
##
## Residual Deviance: 8638.914
## AIC: 8686.914
```

The output gives, Residual Deviance and AIC of model and coefficients of variables. Residual Deviance and AIC can be used in comparisons of nested models, but here, there is only one model so these values won’t be used for this study. The model summary output has a block of coefficients and a block of standard errors parts. Each of these blocks has one row of values corresponding to a model equation. First row of the block of coefficients, comparing **status = "fired"** to our baseline **status = "ok"** and the second row comparing **status = "promoted"** to our baseline **status = "ok"**. In theory, coefficients refer to how much a one-unit increase in the respective variable will increase or decrease the target variable.

A one-unit increase in the variable **age** is associated with the decrease in the log odds of “ok”(not fired and not promoted) vs. get fired(“fired”) in the amount of 0.0014.

A one-unit increase in the variable **age** is associated with the increase in the log odds of being not fired and not promoted(“ok”) vs. to be promoted(“promoted”) in the amount of 0.0015.

-The log odds of not fired and not promoted(“ok”) vs. get fired(“fired”) will decrease by 1.569 if moving from **salary="0"** to **salary="1"**.

-The log odds of not fired and not promoted(“ok”) vs. get fired(“fired”) will decrease by 2.519 if moving from **salary="0"** to **salary="2"**.

-The log odds of not fired and not promoted("ok") vs. get fired("fired") will decrease by 2.403 if moving from **salary="0"** to **salary="3"**.

-The log odds of not fired and not promoted("ok") vs. get prompted("promoted") will increase by 0.211 if moving from **salary="0"** to **salary="1"**.

-The log odds of not fired and not promoted("ok") vs. get prompted("promoted") will increase by 0.526 if moving from **salary="0"** to **salary="2"**.

-The log odds of not fired and not promoted("ok") vs. get prompted("promoted") will increase by 0.124 if moving from **salary="0"** to **salary="3"**.

A few examples are given above for different situations. With the same way, similar interpretations can be made for other situations and other variables. Although the coefficients are obtained, the essential information to test their significance cannot be obtained from here. Since this output does not give us this information, these values will have to be calculated manually.

Significance of The Features

In order to decide whether the coefficients are significant or not, the probability values should be calculated. This process is shown below.

```
#Significance of the features:
z <- summary(model)$coefficients/summary(model)$standard.errors
p <- (1-pnorm(abs(z),0,1))*2
p

##           (Intercept) gendermale       age hours evaluation3 evaluation4
## fired               0  0.4344384 0.6494981     0   0.6883567    0.368659
## promoted            0  0.9470791 0.6823264     0   0.8256798    0.000000
##           evaluation5   salary1   salary2   salary3   salary4   salary5
## fired       0.2623264 0.0000000 0.0000000 0.0000000 0.0000000 0.6111546
## promoted    0.0000000 0.2114853 0.7509396 0.4462154 0.5422735 0.5824307
```

First of all, the hypothesis regarding this result is $H_0: \beta_i = 0,\ H_1: \beta_i \neq 0,\ i = 0,1, \ldots$. If H_0 is not rejected, it means that the coefficient of that variable is not significant. In this case, it means that this variable does not make a significant difference in the model. This decision is made by looking at the p values in the output. If the significance of a categorical variable is discussed, this variable has a significant effect on the target variable if any of its levels are meaningful so in this model, we have 3 categorical variable which are "gender", "evaluation" and "salary".

"gender" is not significant at 0.05 significant level on this model. For "gender" coefficients, H_0 could not rejected. This variable does not have a significant statistical effect on "fired" or "promoted".

"evaluation" is significant at 0.05 significant level on "promoted". H_0 rejected because at least one of them has smaller p-value than 0.05. Thanks to, the "4" and "5" levels of the variable, it has significant statistical effect on "promoted" but for "fired", this variable is not significant at 0.05 significant. H_0 does not rejected because all of them has bigger p-value than 0.05. This variable does not have a significant statistical effect on "fired".

"salary" is significant at 0.05 significant level on "fired". H_0 rejected because at least one of them has smaller p-value than 0.05. Thanks to, the "1", "2", "3" and "4" levels of the variable, it has significant statistical effect on "fired" but for "promoted", this variable is not significant at 0.05 significant. H_0 does not rejected because all of them has bigger p-value than 0.05. This variable does not have a significant statistical effect on "promoted".

“age” is not significant at 0.05 significant level on this model. All p-values are greater than 0.05. H_0 could not rejected for both “fired” or “promoted”. Variable does not has a significant statistical effect on the “fired” or “promoted”.

“hours” is significant at 0.05 significant level on this model. All p-values are smaller than 0.05. H_0 rejected for both “fired” or “promoted”. Variable has a significant statistical effect on the “fired” or “promoted”.

Relative Risk Ratio

The ratio of the probability of choosing one outcome category over the probability of choosing the baseline category is often referred as relative risk (and it is sometimes referred to as *odds*, described in the regression parameters above). The exponentials of the model's coefficients can be taken to see these risk ratios.

```
# exponentiate of coefficients
exp(coef(model))

##          (Intercept) gendermale       age     hours evaluation3 evaluation4
## fired    2.045212e+02  0.9456219 0.9985956 0.9226451   1.0336498    1.113944
## promoted 5.018361e-04  0.9944795 1.0014773 1.1193802   0.9748307   35.687687
##          evaluation5   salary1    salary2   salary3  salary4  salary5
## fired       1.142907 0.2083294 0.08057238 0.0904168 0.171700 1.080718
## promoted   30.484338 1.2344062 1.05401440 1.1342642 1.109579 1.118016
```

-The relative risk ratio for a one-unit increase in the variable **“hour”** is 0.923 for “ok”(not fired and not promoted) vs. get fired(“fired”)

-The relative risk ratio for a one-unit increase in the variable **“age”** is 0.998 for “ok”(not fired and not promoted) vs. get fired(“fired”)

-The relative risk ratio switching from **salary="0"** to **salary="1"** is 0.208 for “ok”(not fired and not promoted) vs. get fired(“fired”)

-The relative risk ratio switching from **evaluation="2"** to **salary="3"** is 0.975 for “ok”(not fired and not promoted) vs. get promoted(“promoted”)

A few examples are given above for different situations. With the same way, similar interpretations can be made for other situations and other variables.

Predicting Probabilities of The Target Variable & Model Performance

To predict the probabilities, `predict()` function whit “type=probs”. The first 6 probability values displayed below.

```
#Predicted Probablities of the target variable:
predicted_probs <- predict(model,type = "probs")
head(predicted_probs)

##             ok      fired    promoted
## 636  0.4700945 0.51576975 0.014135759
## 9278 0.2880603 0.70217970 0.009759971
## 8869 0.6725894 0.25847358 0.068937030
## 6316 0.1151084 0.87973719 0.005154429
## 6642 0.6059032 0.08076231 0.313334536
## 9720 0.1434931 0.04539913 0.811107723
```

Here the prediction is determined according to the highest probability. Since "fired" has the highest probability value for the 636th observation in the first row, we will assume that this observation predicts as "fired". This process will continue for other predicted observations. To compare these predictions with the test, we need to rename them. “apply” function will be used for this action. The renaming process and the performance of the model are calculated as follows

```
#Performance of the Model on Train and Test Set:
#For TRAIN
predicted_probs_train <- predict(model, type = "probs")
predicted_class_train <- colnames(predicted_probs_train)[apply(predicted_probs_train, 1,whi
ch.max)]
print(c("Train:", mean(predicted_class_train == train$status)))

## [1] "Train:"           "0.68153576549307"

#For TEST
predicted_probs_test <- predict(model, test, type = "probs")
predicted_class_test <- colnames(predicted_probs_test)[apply(predicted_probs_test, 1, which
.max)]
print(c("Test:", mean(predicted_class_test == test$status)))

## [1] "Test:"             "0.696178343949045"
```

The predicted probabilities are renamed so `predicted_class_train` is a vector now and the elements of this vector are “ok”, “fired” or “promoted”. Now we are ready to calculate performance the model. This ratio represents, model's performance on “train”. The model classifies the “train” approximately %68 correctly. We did the same process that we did with "train" before to testing data. The accuracy ratio that we obtained using the “train” is smaller than “test”. We might suspect of underfitting problem. Getting more training data, increasing the size or number of parameters in the model or increasing the complexity of the model may decrease or fix this problem. The higher this test performance ratio means; the better estimation result we get. The model classifies the “testing” approximately 70% correctly. Although this value is high, it would be wrong to say that it failed or succeed before checking the confusion matrix. For final decision, let’s construct confusion.

```
#Confusion Matrix of The Model
confmatr <- table(predicted = predicted_class_test, actual = test$status)
confmatr

##            actual
## predicted  fired  ok promoted
##    fired     425 118       52
##    ok        107 202       49
##    promoted   62  89      466



# Accuracy of The Model
acc <- sum(diag(confmatr)) / sum(confmatr)
acc

## [1] 0.6961783
```

Accuracy of the model represents ratio of correct predictions to total observations count. This ratio is low for this study. This ratio’s interpretation actually depends on the content of the analysis and the Analyzer. In my opinion, 30% tolerance cannot be ignored as it relates to layoffs. For example, according to confusion matrix 169/594≈%33 of employee who should be fired will not fire and 62 of them will get promotion. Also 101/567≈%18 of employee who should get promotion will not get promotion and 52 of them will be fired. These are examples of undesirable situations. In this case, for someone who thinks this way, it would not be appropriate to use this model.