Multinomial Logistic Regression Model - Decision Tree (Classification)

Merve Topdemir

16 04 2021

The data to be used in this study is " HR: Human Resources Data" from the "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. Decision Tree and Multinomial Logistic Regression models will be created on this data set and the performances of these models will be compared.

```
#install.packages("DALEX")
library(DALEX)

data <- DALEX::HR
data$evaluation <-as.factor(data$evaluation)
data$salary <-as.factor(data$salary)
head(data)
```

```
##      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
```

```
table(data$status)
```

```
##
##    fired     ok   promoted
##     2855   2221       2771
```

Data Structure

The data set contains 7847 observations and 6 variables.

Dependent (target) variable:

- **status** : Heart disease (fired – promoted – ok)

Independent (feature) variables:

- **gender** - gender of an employee.
- **age** - age of an employee in the moment of evaluation.
- **hours** - average number of working hours per week.
- **evaluation** - evaluation in the scale 2 (bad) - 5 (very good).
- **salary** - level of salary in the scale 0 (lowest) - 5 (highest).

```
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)
```

```
##     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
##
```

Factors are used to represent categorical data. Numerical variable is one that may take on any value within a finite or infinite interval.

- Status is a factor variable, it has 3 levels, 2855 **fired**, 2221 **ok** and 2771 **promoted**.
- Gender is a factor variable, it has 2 levels, 3949 **female** and 3898 **male**.
- Evaluation is a factor variable, it has 4 levels, 2371**(2)** , 2272**(3)**, 1661**(4)** and 1543**(5)**.
- Evaluation is a factor variable, it has 6 levels, 1105**(0)** , 1417**(1)**, 1461**(2)**, 1508**(3)**, 1316**(4)** and 1040**(5)**.
- Age variable minimum value is 20, maximum value is 60, median is 40.16, mean is 40, first quartile is 30.03 and third quartile is 49.96.
- Hours minimum value is 35, maximum value is 79.98, median is 46.28, mean is 49.71, first quartile is 37.64 and third quartile is 59.48.

Splitting Data (Create train/test set)

The following code splits 80% of the data selected randomly into training set and the remaining 20% sample into test set.

```
set.seed(123)
index <- sample(nrow(data),nrow(data)*0.8)
train <- data[index,]
test <- data[-index,]
table(train$status);table(test$status)

##
##    fired       ok promoted
##     2322     1716     2239

##
##    fired       ok promoted
##      533      505      532
```

Training Multinomial Logistic Regression Model

```
#Training a MLRM
train$status <- relevel(train$status, ref = "ok")
#install.packages("nnet")
library(nnet)

model <- multinom(status ~ ., data =train , trace = FALSE)
summary(model)
```

```
## Call:
## multinom(formula = status ~ ., data = train, trace = FALSE)
##
## Coefficients:
##          (Intercept)   gendermale       age          hours    evaluation3
## fired       5.690161  -0.042647182  -0.002966112  -0.08641021   0.1061064
## promoted   -7.890035   0.002364242   0.002793503   0.11684543   0.1188223
##          evaluation4  evaluation5  salary1     salary2       salary3    salary4
## fired      0.1157423   0.109052   -1.5324575  -2.54253620  -2.3967894  -1.7223920
## promoted   3.5985066   3.503691    0.2406319   0.07713669   0.0763731   0.1688317
##              salary5
## fired    -0.05609585
## promoted -0.04157942
##
## Std. Errors:
##          (Intercept)   gendermale       age       hours    evaluation3 evaluation4
## fired      0.2745504  0.07213380   0.003143139 0.004171817  0.08366726   0.1189806
## promoted   0.3591567  0.08427709   0.003658093 0.004255995  0.11625759   0.1408463
##          evaluation5   salary1   salary2   salary3   salary4   salary5
## fired      0.1206026 0.1362585 0.1389622 0.1373626 0.1367832 0.1515054
## promoted   0.1413705 0.1697613 0.1654981 0.1652011 0.1710001 0.1992210
##
## Residual Deviance: 8501.111
## AIC: 8549.111
```

Predicted Probabilities of the Target Variable

```
predicted_probs <- predict(model, type = "probs")
head(predicted_probs)
```

```
##              ok      fired    promoted
## 3106 0.32024742  0.66883805  0.010914527
## 3162 0.20007393  0.09990864  0.700017433
## 2811 0.22298622  0.01878187  0.758231910
## 673  0.32772966  0.29273531  0.379535023
## 5442 0.11017188  0.88558026  0.004247866
## 3779 0.06980798  0.87236133  0.057830683
```

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,which.max)]
print(c("Train:", mean(predicted_class_train == train$status)))
```

```
## [1] "Train:"            "0.688386171738091"
```

```
#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:"             "0.661146496815287"
```

According to the result, the response variables in the test data are classified correctly by 68% and the train data is classified correctly by 66%.

Confusion Matrix

```
#install.packages("e1071")
library(e1071)

#install.packages("caret")
library(caret)

confusionMatrix(
factor(predicted_class_test, levels = c("fired","ok","promoted")),
factor(test$status, levels = c("fired","ok","promoted"))
)
```

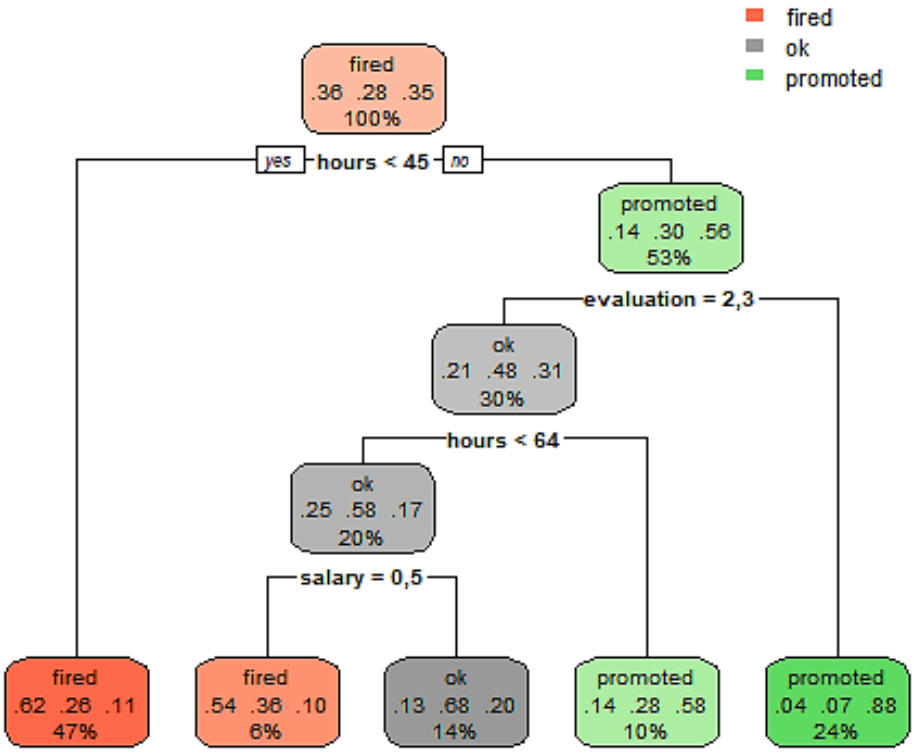
```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction fired  ok promoted
##   fired      392 176       58
##   ok          78 219       47
##   promoted    63 110      427
##
## Overall Statistics
##
##                Accuracy : 0.6611
##                  95% CI : (0.6371, 0.6846)
##     No Information Rate : 0.3395
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4902
##
##  Mcnemar's Test P-Value : 1.16e-13
##
## Statistics by Class:
##
##                      Class: fired  Class: ok   Class: promoted
## Sensitivity                0.7355    0.4337          0.8026
## Specificity                0.7743    0.8826          0.8333
## Pos Pred Value             0.6262    0.6366          0.7117
## Neg Pred Value             0.8506    0.7667          0.8918
## Prevalence                 0.3395    0.3217          0.3389
## Detection Rate             0.2497    0.1395          0.2720
## Detection Prevalence       0.3987    0.2191          0.3822
## Balanced Accuracy          0.7549    0.6581          0.8180
```

Decision Tree

```
#install.packages("rpart")
library(rpart)

#install.packages("rpart.plot")  # Plot the tree
library(rpart.plot)
```

```
model <- rpart(status ~ ., data = data, method = "class")
rpart.plot(model)
```



The colors in the decision tree graph and the values written at the top of the boxes represent the status variable. The ratios in the boxes show the status "fired", "ok", "promoted" ratios there, respectively. The rate at the bottom of the boxes shows the rate at which the data is there. For example, the top box (root node) contains 0.36 fired, 0.28 ok, 0.35 promoted, and it writes 100% because the data 13 has never been split. Below is the first partitioning process, if the hours is less than 45, it means move to the yes side, if not to the side that says no. In the first partitioning operation, when the hours feature is less than 45, it goes to the leftmost box. Box on the left, 0.62 has the value "fired", 0.27 has "ok", 0.12 has "promoted". In total, 47% of the data is here. In summary, 47% of the data work less than 45 hours, and the model predicts them as status "fired". Other nodes can be interpreted similarly.

Splitting Data (Create train/test set)

```
# Data Splitting
set.seed(123)
index <- sample(nrow(data), nrow(data) * 0.8)
train <- data[index,]
test <- data[-index,]


table(train$status)

##
##    fired       ok promoted
##     2322     1716     2239

table(test$status)

##
##    fired       ok promoted
##      533      505      532
```

Performance of the Model on Train and Test Set

```
model <- rpart(status ~., method = "class", data = train)


#For train
pred_labels_train <- predict(model, train, type = "class")
conf_mat_train <- table(pred_labels_train, train$status)
print(c("Train:",sum(diag(conf_mat_train))/sum(conf_mat_train)))

## [1] "Train:"            "0.691891030747172"

#For test
pred_labels_test <- predict(model, test, type = "class")
conf_mat_test <- table(pred_labels_test, test$status)
print(c("Test:",sum(diag(conf_mat_test))/sum(conf_mat_test)))

## [1] "Test:"             "0.642675159235669"
```

According to the result, the response variables in the test data are classified correctly by 64% and the train data is classified correctly by 69%.

Confusion Matrix

```
confusionMatrix(
factor(pred_labels_test, levels = c("fired","ok","promoted")),
factor(test$status, levels = c("fired","ok","promoted"))
)
```

```
## Confusion Matrix and Statistics
##           Reference
## Prediction fired  ok promoted
##   fired      465 248      113
##   ok          28 167       42
##   promoted    40  90      377
##
## Overall Statistics
##                Accuracy : 0.6427
##                  95% CI : (0.6184, 0.6664)
##     No Information Rate : 0.3395
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4614
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: fired  Class: ok  Class: promoted
## Sensitivity                0.8724    0.3307           0.7086
## Specificity                0.6519    0.9343           0.8748
## Pos Pred Value             0.5630    0.7046           0.7436
## Neg Pred Value             0.9086    0.7464           0.8542
## Prevalence                 0.3395    0.3217           0.3389
## Detection Rate             0.2962    0.1064           0.2401
## Detection Prevalence       0.5261    0.1510           0.3229
## Balanced Accuracy          0.7622    0.6325           0.7917
```

	Performance (Accuracy)
Multinomial LRM (Train)	0.68
Multinomial LRM (Test)	0.66
Decision Tree (Train)	0.69
Decision Tree (Test)	0.64

The decision tree model has an accuracy of 0.64, and the multinomial logistic regression model has an accuracy of 0.66. The multinomial logistic regression model is more successful on predicting employees of status.