Tutorial 10 - Cross-Validation & Evaluation

AUTHOR
Victoria Hünewaldt

Tutorial 10

This tutorial will cover out-of-sample predictions, cross-validatoin and evaluation metrics. You will learn:

- how to split your data into a test & training set
- how to train your model on the training data & make predictions on the test data
- how to calculate the out-of-sample RSME
- how to apply a cross-validation procedure
- how to get a confusion matrix & evaluate your model's performance based on accuracy, precision, recall

Exercises

You will need to use the "caret" library. Create a .qmd-file and solve the tasks there. Store it in the JupyterHub folder "Session 10".

Out-of-sample prediction

Use your results from the logistic regression model from tutorial 9. Test how your model performs out-of-sample. Perform the following steps.

1. Split your data into a training and test sample (70% training, 30% testing).
2. Train your model on the training data.
3. Make predictions on the test set using the trained model.
4. Calculate the out-of-sample RMSE, i.e. the test set RMSE.

```
# clean environment
rm(list=ls())

# load packages/libraries

# install.packages("remotes") # hashtagged because already installed
library(remotes)
remotes::install_gitlab("BAQ6370/sozoekds", host="gitlab.rrz.uni-hamburg.de")
library(sozoekds)

library(tidyverse)
library(dplyr)
library(caret) # for splitting, CV
```

```r
airbnb_data <- airbnbsmall # store data as "airbnb_data"

##################

# 1. split data into training and test set

# Define the x-variables of your model
xvar <- c("n_number_of_reviews", "price", "d_gym") # creates a column w/ those 3 variables; "s
x <- airbnb_data[, xvar] # defines x as the variables in xvar

# Define the output variable of your model
airbnb_data$high_rating = ifelse(airbnb_data$n_review_scores_rating>94, 1, 0) # create high ra
y <- airbnb_data$high_rating # defines y as "high_rating" in the airbnb dataset

# Split the data into training and testing sets (70% training, 30% testing)
set.seed(123)  # for reproducibility; to get the same random split each time you run your code

# creates the training data
trainIndex <- createDataPartition(y, p = 0.7, # percentage of data going to training
                                  list = FALSE,
                                  times = 1) # only 1 split

x_train <- x[trainIndex, ]
x_test <- x[-trainIndex, ]
y_train <- y[trainIndex]
y_test <- y[-trainIndex]


# 2. Train your model on the training data
model <- glm(y_train ~ ., family=binomial(link="logit"), data = data.frame(cbind(y_train, x_tr
summary(model)

# 3. Make predictions on the test set using the trained model
out_predictions <- predict(model, newdata = data.frame(x_test))

# 4. Calculate the out-of-sample RMSE, i.e. the test set RMSE.
residuals <- (y_test-out_predictions) # calculate the error (predicted values-observed values
sqrt(mean(residuals^2)) # prints 0.772622
```

Cross-validation, confusion matrix & evaluation metrics

Now apply the cross-validation procedure using the training data. Use the “trainControl” command and do a 5-fold and 10-fold cross-validation. Compare the results.

```r
### 5-fold CV ###

set.seed(123) # for reproducability

# specify cross-validation & number of folds
cross <- trainControl(method="cv", # cross-validation
                      number=5, # k=5
                    verboseIter = TRUE)
```

```
# specify model to be estimated using training data & apply cv
modelcross <- train(y_train ~ ., data=data.frame(cbind(y_train, x_train)),
                method="glm",
                family=binomial,
                trControl=cross) # applies our cross-validation process as specified above
modelcross # prints RMSE 0.4941984

### 10-fold CV ###

set.seed(123) # for reproducability

# specify cross-validation & number of folds
cross <- trainControl(method="cv", # cross-validation
                          number=10, # k=10
                      verboseIter = TRUE)

# specify model to be estimated using training data & apply cv
modelcross <- train(y_train ~ ., data=data.frame(cbind(y_train, x_train)),
                method="glm",
                family=binomial,
                trControl=cross) # applies our cross-validation process as specified above
modelcross # prints RMSE 0.4946834
```

To evaluate your model create a confusion matrix with the “confusionMatrix” command and find accuracy, precision and recall scores.
For the confusion matrix you will need two classes “negative” and “positive” of the predicted values as well as two classes “negative” and “positive” of the observed values from the test set. Predicted probabilities <0.5 should fall in the predicted class “negative” (0), whereas predicted probabilities >0.5 should fall in the predicted class “positive” (1).

```
# create the classes of the predicted values and store in "p_class"
p_class <- factor(ifelse(out_predictions > 0.5, 1, 0), levels = c(0, 1), labels = c("Negative_

# levels in predicted values and observed values need to be the same, but are different:
unique(levels(p_class)) # prints "Negative_Class" "Positive_Class"
unique(levels(y_test)) # prints NULL

# therefore transform y_test variables into factor w/ same labels as p_class
y_test_f <- factor(y_test, levels = c(1, 0), labels = c("Positive_Class", "Negative_Class"))

confusionMatrix(p_class, y_test_f, mode = "prec_recall" ) # prints confusion matrix
```