Tutorial 12 - Tree-based Models

AUTHOR

Victoria Hünewaldt

Tutorial 12

This tutorial will cover decision trees.

You will learn:

- how to run a regression decision tree
- how to run a classification decision tree
- how to visualise decision trees
- how to evaluate its performance on test & training data

Exercises

Use the airbnbsmall data set. You will need to use the “rpart” and “rpart.plot” library. Create a .qmd-file and solve the tasks there. Store it in the JupyterHub folder “Session 12”.

Regression Decision Tree

Run a regression decision tree explaining the variable “price” (= endogenous variable). Use all other variables as potential predictor variables (i.e. specify a full model).

Print and plot the tree.

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

# load packages/libraries

#install.packages("rpart")
library(rpart) # for creating trees
#install.packages("rpart.plot")
library(rpart.plot) # for plotting trees
#remotes::install_gitlab("BAQ6370/sozoekds", host="gitlab.rrz.uni-hamburg.de")
library(sozoekds)
library(dplyr)

#load data
airbnb_data <- airbnbsmall # store data as "airbnb_data"

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

# 1. Regression tree
```

```
regtree <- rpart(
  formula = price~.,
  data=airbnb_data,
  method = "anova"
)

# print
regtree
# plot
rpart.plot(regtree)
```

Classification Decision Tree

Split your data into test and training data. Run a classification decision tree explaining the variable “high_rating” (= endogenous variable) on the training data. Use all other variables as potential predictor variables (i.e. specify a full model).

Create a confusion matrix for the training data and one for the test data.

Compare your results to the results obtained by logistic regression in tutorial 10.

```
rm(list=ls())

# load packages/libraries

#install.packages("rpart")
library(rpart) # for creating trees
#install.packages("rpart.plot")
library(rpart.plot) # for plotting trees
#remotes::install_gitlab("BAQ6370/sozoekds", host="gitlab.rrz.uni-hamburg.de")
library(sozoekds)
library(dplyr)
library(caret) # for splitting

# load data
airbnb_data <- airbnbsmall # store data as "airbnb_data"

# binary variable "high rating"

airbnb_data$high_rating = ifelse(airbnb_data$n_review_scores_rating>94, 1, 0)
airbnb_data_2 <- select(airbnb_data, -n_review_scores_rating)

y <- airbnb_data_2$high_rating # defines y as "high_rating" in the airbnb dataset

# split data
set.seed(123)  # for reproducibility; to get the same random split each time you run your code

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

```r
train <- airbnb_data_2[trainIndex,]
test <- airbnb_data_2[-trainIndex,]

# train the tree
classtree <- rpart(
  formula = high_rating~.,
  data=train,
  method = "class",
  )

# (plot the tree)
#rpart.plot(classtree)
#classtree


# training confusion matrix
train_predict <- predict(classtree, data=train, type="class")

tab1 <- table(predict = train_predict, actual = train$high_rating)

confusionMatrix(tab1, mode = "prec_recall")


# testing confusion matrix

test_predict <- predict(classtree, newdata=test, type="class")

tab2 <- table(predict = test_predict, actual = test$high_rating)

confusionMatrix(tab2, mode = "prec_recall")
```