Exercises - random forests

Jan-Philipp Kolb

27 Mai, 2019

Exercise: random forests

Download and import example data

1) Download the Data from **here** and read in the `adult.csv` file with `header=FALSE`. Store this in df. Use `str()` command to see the dataframe.
2) Get the column names from the **meta data** and add them to the data frame. Notice the df is ordered - V1,V2,V3,... and so on.

Get an overview of the data

3) Use the `table` command to get the distribution of the `class` feature.
4) Make a binary variable `class`.
5) Use the `cor()` command to see the corelation of all the numeric and integer columns including the class column.

Solution: Download and import (I)

Exercise 1

```
l1 <- "http://www.r-exercises.com/wp-content"
l2 <-"/uploads/2016/11/adult.csv"
link <- paste0(l1,l2)
df <- read.csv(link,header=FALSE)
str(df)
```

```
## 'data.frame':    15916 obs. of  15 variables:
##  $ V1 : int  39 50 38 53 28 37 49 52 31 42 ...
##  $ V2 : Factor w/ 9 levels " ?"," Federal-gov",..: 8 7 5 5 5
##  $ V3 : int  77516 83311 215646 234721 338409 284582 160187 2
##  $ V4 : Factor w/ 16 levels " 10th"," 11th",..: 10 10 12 2 10
##  $ V5 : int  13 13 9 7 13 14 5 9 14 13 ...
##  $ V6 : Factor w/ 7 levels " Divorced"," Married-AF-spouse",.
##  $ V7 : Factor w/ 15 levels " ?"," Adm-clerical",..: 2 5 7 7
##  $ V8 : Factor w/ 6 levels " Husband"," Not-in-family",..: 2
##  $ V9 : Factor w/ 5 levels " Amer-Indian-Eskimo",..: 5 5 5 3
##  $ V10: Factor w/ 2 levels " Female"," Male": 2 2 2 2 1 1 1 2
```

## Solution: Download and import (II)

Exercise 2

```
colnames(df) <- c("age","workclass","fnlwgt","education",
                 "education-num","marital-status","occupation",
                 "relationhip","race","sex","capital-gain",
                 "capital-loss","hours-per-week",
                 "native-country","class")
```

Solution: Get overview

Exercise 3

```
table(df$class)
```

```
##
##  <=50K   >50K
##  12097   3819
```

Exercise 4

```
df$class <- ifelse(df$class==" >50K", 1, 0)
```

Exercise 5

```
cor(df[,c(1,3,5,11,12,13,15)])
```

```
##                        age       fnlwgt education-num capital
## age             1.00000000 -0.079506361    0.02668698 0.0664
## fnlwgt         -0.07950636  1.000000000   -0.04671504 0.0006
## education-num   0.02668698 -0.046715043    1.00000000 0.1174
## capital-gain    0.06646649  0.000653693    0.11745307 1.0000
```

Exercise: random forests

Split the dataset

6) Split the dataset into Train and Test sample. You may use `caTools::sample.split()` and use the ratio as 0.7 and set the seed to be 1000.
7) Check the number of rows of Train and Test
8) We are ready to use decision tree in our dataset. Load the package `rpart` and `rpart.plot`
9) Use `rpart` to build the decision tree on the Train set. Include all features. Store this model in `dec`
10) Use `prp()` to plot the decision tree.

6

```
set.seed(1000)
library(caTools)
split=sample.split(df$class, SplitRatio=0.8)
Train=df[split==TRUE,]
Test=df[split==FALSE,]
```

Exercise

1) use the `predict()` command to make predictions on the Train data. Set the method to `class`. Class returns classifications instead of probability scores. Store this prediction in pred_dec.
2) Print out the confusion matrix
3) What is the accuracy of the model. Use the confusion matrix.
4) What is the misclassification error rate? Refer to Basic_decision_tree exercise to get the formula.
5) Lets say we want to find the baseline model to compare our prediction improvement. We create a base model using this code

```
length(Test$class)
```

```
## [1] 3183
```

```
base=rep(1,3183)
```

- Use the table() command to create a confusion matrix between the base and Test$class.

6) What is the number difference between the confusion matrix accuracy

Solution

```
library(caTools)
colnames(df)=c("age","workclass","fnlwgt","education","education
df$class=ifelse(df$class==" >50K", 1, 0)
df$class=as.factor(df$class)
set.seed(1000)

split=sample.split(df$class, SplitRatio=0.8)

Train=df[split==TRUE,]

Test=df[split==FALSE,]
library(rpart)
library(rpart.plot)
dec=rpart(class~., data=Train)
par(mar = rep(2, 4))
```

1)

```
pred_dec=predict(dec,newdata = Test,type="class")
```