Exercises - random forests

Jan-Philipp Kolb

31 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 `dat`. 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 `dat` 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)

Solution Exercise 1: get the dataset

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

str(dat)

## '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
```

Solution: Download and import (II)

Solution Exercise 2 rename columns

```
colnames(dat) <- 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

Solution Exercise 3: get distribution

```
table(dat$class)
```

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

Solution Exercise 4

- A binary variable class

```
levels(dat$class) <- c(0,1)
dat$class <- as.numeric(dat$class)
```

Solution Exercise 5

Correlation of all numeric and integer variables

```
cor(dat[,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
## capital-loss  0.06176551 -0.012139341    0.08090257 -0.0316
## hours-per-week 0.05659864 -0.012345724    0.14528405  0.0757
## class         0.22920766 -0.013067759    0.32856870  0.2210
##                capital-loss hours-per-week       class
## age             0.06176551     0.05659864  0.22920766
## fnlwgt         -0.01213934    -0.01234572 -0.01306776
## education-num   0.08090257     0.14528405  0.32856870
## capital-gain   -0.03168533     0.07571567  0.22104995
## capital-loss    1.00000000     0.05439109  0.15366554
## hours-per-week  0.05439109     1.00000000  0.22544319
```

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.

Solutions

Solution Exercise 6

```
dat$class <- as.factor(dat$class)
```

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

Solution Exercise 7

```
nrow(Train)
```

```
## [1] 12733
```

```
nrow(Test)
```

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

Solution Exercise 8

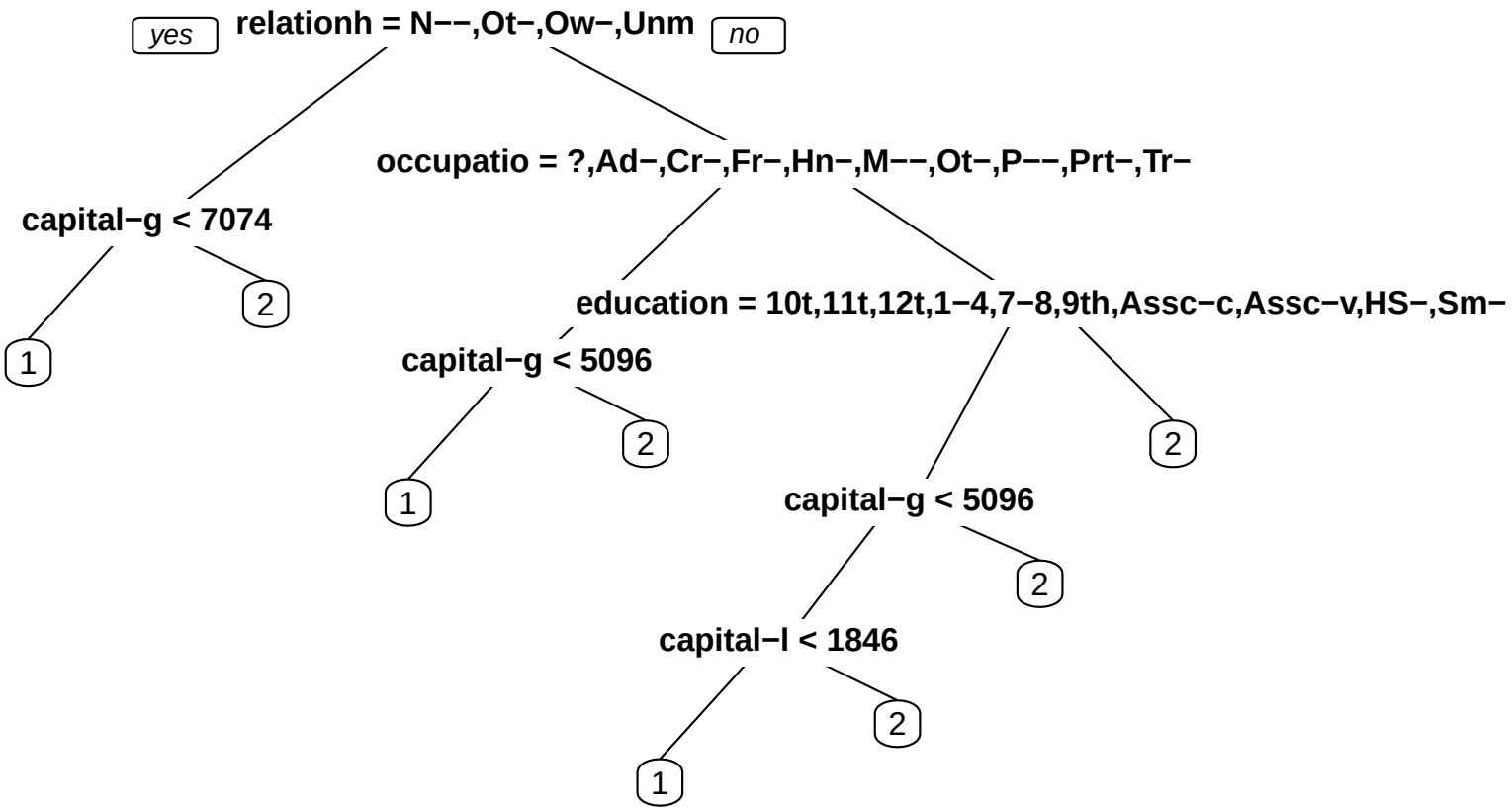
```
library(rpart)
library(rpart.plot)
```

Solution Exercise 9

```
dec <- rpart(class~., data=Train)
```

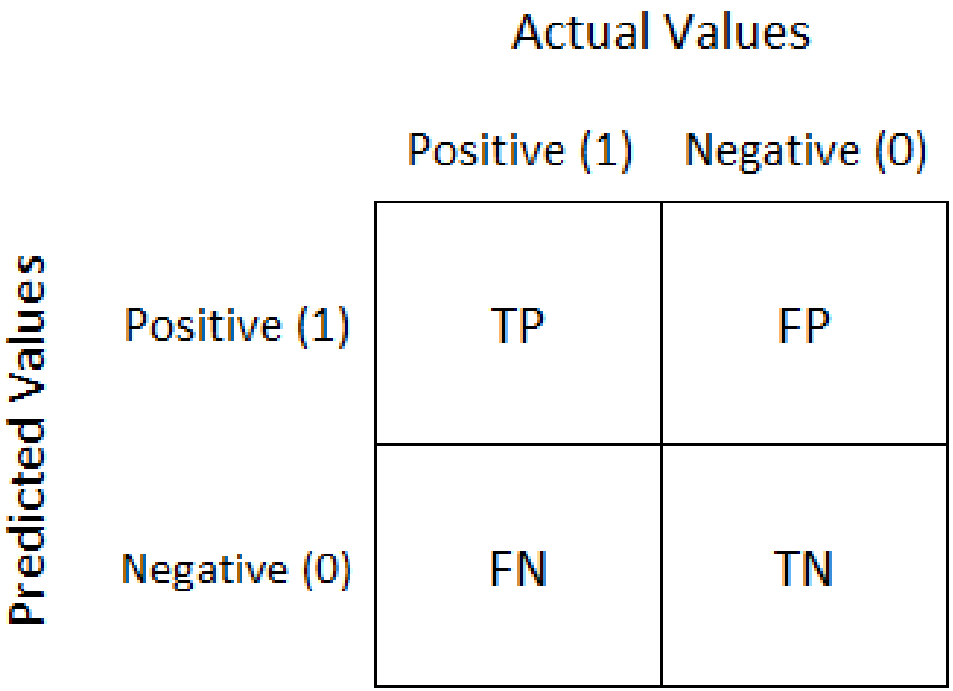
Solution Exercise 10

```
par(mar = rep(2, 4))
prp(dec)
```



Exercise - predict and confusion matrix

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**



Exercises - accuracy

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.

Exercises

7) Remember the `predict()` command in question 1. We will use the same mode and same command except we will set the method to "regression". This gives us a probability estimates. Store this in pred_dec_reg

8) load the `ROCR` package.

Use the `prediction()`, `performance()` and `plot()` command to print the ROC curve. Use `pred_dec_reg` variable from Q7. You can also refer to the previous exercise to see the code.

9) `plot()` the same ROC curve but set `colorize=TRUE`

10) Comment on your findings using ROC curve and accuracy. Is it a good model? Did you notice that ROC `prediction()` command only takes probability predictions as one of its arguments. Why is that so?

Solution Exercise 1

Split the dataset

```
split <- caTools::sample.split(Y = dat$class, SplitRatio=0.7)
Train <- dat[split==TRUE,]
Test <- dat[split==FALSE,]

library(rpart)
library(rpart.plot)

dec <- rpart(class~., data=Train)

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

Solutions

Exercise 2 - Confusion matrix

```
table(Test$class,pred_dec)
```

```
##    pred_dec
##        1    2
##   1 3427  202
##   2  556  590
```

Solution Exercise 3

```
(2345+364)/(2345+364+400+74)
```

```
## [1] 0.8510839
```

Solution Exercise 4

```
mean(as.factor(Test$class)!=pred_dec)
```

```
## [1] 0.1587435
```

Solution Exercise 5

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

```
## [1] 4775
```

```
base <- rep(1,nrow(Test))
table(Test$class,base)
```

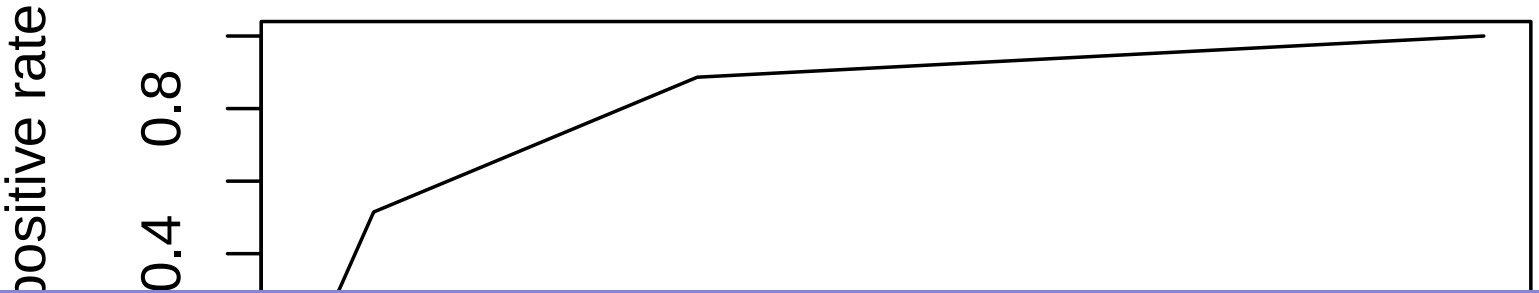
Further solutions

Solution Exercise 7

```
pred_dec_reg <- predict(dec,newdata = Test,type="prob")
```

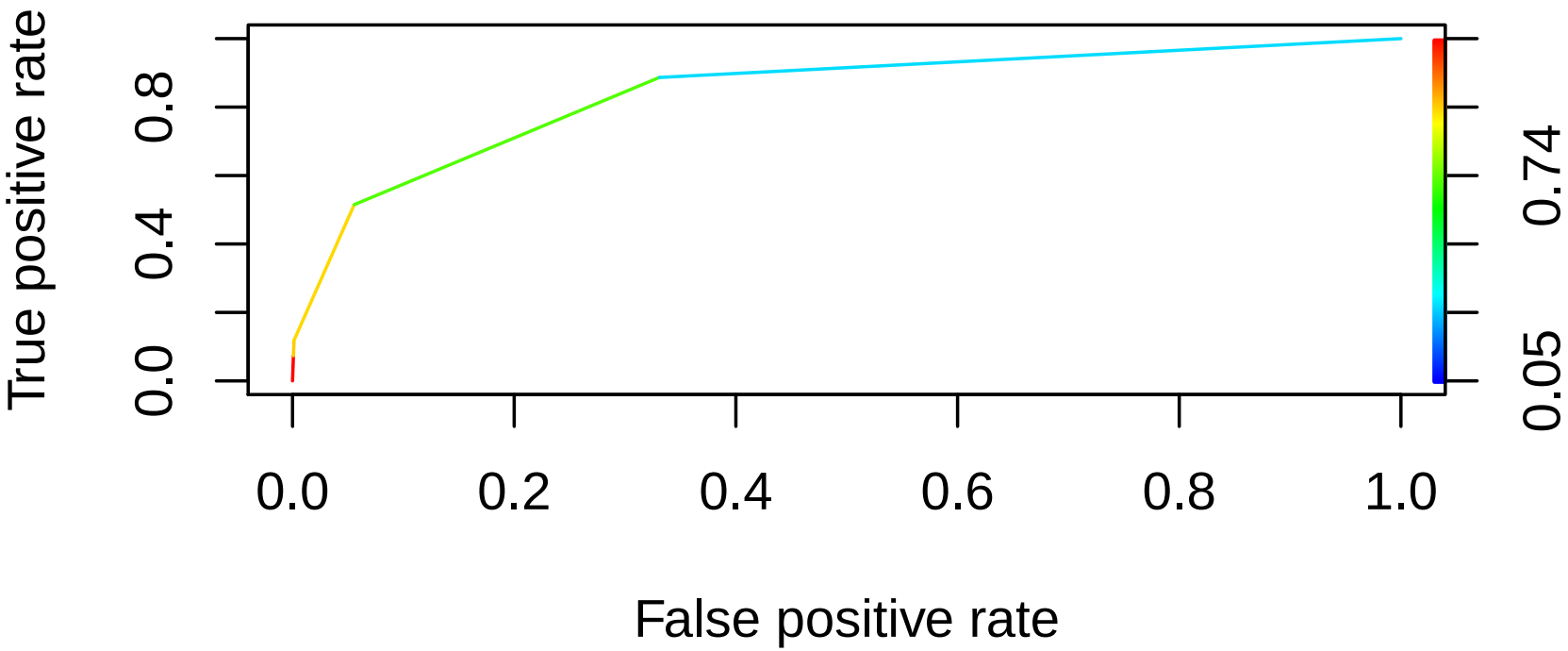
Solution Exercise 8

```
library(ROCR)
pred <- prediction(predictions=as.numeric(pred_dec_reg[,2]),Test
perf <- performance(pred,"tpr","fpr")
plot(perf)
```



Solution Exercise 9

```
plot(perf,colorize=TRUE)
```



Solution Exercise 10

- It is a good model. The initial accuracy is 0.85 which is pretty good.
- The ROC curve is also leaning more towards the true positive side which is also a good sign. ROC `prediction()` command takes probability score predictions because it is used to give a visual representation of a range of threshold values.
- We can use ROC also to interpret what threshold value to chose and decide the ratio of true positive to false positives based on the problem at hand. That is for another exercise