Activity_data

Corine Meinema

11 november 2018

Summary

The data of measurements of some excersize has been studied. The aim is to predict from the data which classe of excersize has been done. From the given urls there is a train and test set. I split the train set to a training and testing set, so I could compare several models. The random forest could perfectly (100%) predict the classe of the testing set.

Importing data

The data can be found here:

```
# set the working directory
setwd("C:/Users/corin/Desktop/coursera/course8/week4")
# the urls of the train and test data
url_train <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
url_test <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

# downloading of the train and test set
download.file(url_train, destfile= ".train.csv")
download.file(url_test, destfile= ".test.csv")

# and read in the train and test data
train <- read.csv(".train.csv")
test <- read.csv(".test.csv")
```

Data

Looking at the data, the test data is very small, only 20 rows. Some of the columns are empty for the test data, and these will give errors on the models. I throw away all the variables where the test data has no value. This might not the most beautiful way to threat the data, but practical.

```
# The selection of the columns of the test and train set where not all the data of
test is na:
train <- train[colSums(!is.na(test)) > 0]
test <- test[colSums(!is.na(test)) > 0]
```

The train data is split in a training and testing set, so I can see how well the model is doing. I used 70% training data and 30% testing data.

```
# set a seed
set.seed(1)

library(caret)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggplot2
```

```
# selection of 70% training and  30% testing data
n = nrow(train)
selection = sample(1:n, size = round(0.7*n), replace=FALSE)
training = train[selection, ]
testing = train[-selection, ]
```

Modeling

I trained the training data to predict the training data later on. Here are the random forest (rf), k-nearest neabour (knn) and a simple decision tree (rpart).

```
# training with random forest, this might take a long time
myrf <- train(classe ~ ., data = training, method = "rf", na.action = na.exclude)

# training with knn
myknn <- train(classe ~ ., data = training, method = "knn", na.action = na.exclude)

# training with rpart
mytree <- train(classe ~ ., data = training, method = "rpart", na.action = na.exclu
de)
```

Predictions

With the models I predict the classe of the testing dataset:

```
outcome_rf <- predict(myrf, testing)
outcome_knn <- predict(myknn, testing)
outcome_tree <- predict(mytree, testing)
```

Outcome of the models

The three models are compared to the classe of the testing data.

```
table(outcome_rf, testing$classe)
```

```
##
## outcome_rf    A    B    C    D    E
##          A 1666    0    0    0    0
##          B    0 1140    0    0    0
##          C    0    0 1082    0    0
##          D    0    0    0  966    0
##          E    0    0    0    0 1033
```

```
table(outcome_knn, testing$classe)
```

```
##
## outcome_knn    A    B    C    D    E
##           A 1518  176    0    0    0
##           B  143  789  192    4    0
##           C    5  174  720  193    3
##           D    0    1  166  578  242
##           E    0    0    4  191  788
```

```
table(outcome_tree, testing$classe)
```

```
##
## outcome_tree    A    B    C    D    E
##            A 1665    0    0    0    0
##            B    1 1140    0    0    0
##            C    0    0    0    0    0
##            D    0    0    0    0    0
##            E    0    0 1082  966 1033
```

It looks like the random forest works can predict the classe of the testing data perfectly. The downside is that this model takes a few houres to calculate.

Outcome of the test data

The outcome of the test data is:

```
predict(myrf, test)
```

```
##  [1] A A A A A A A A A A A A A A A A A A A A
## Levels: A B C D E
```

Remarks

It looks I can predict the testing dataset perfectly with a random forest. There are a few remarks:

- The random forest method takes a lot of time. It is ok to do this once, but for some purposes this is not so useful.
- It is not so nice to throw away all the variables in the test set that are NA, and then make the model on it. On the other side: in the ideal world the test and training set are randomly chosen, so then there should not be a difference in training and test set.
- I did not pass the coursera test. I don't understand what is going wrong: I have split the train data to a training and testing set and made a model independently of the testing set. I could perfectly predict the classe of the testing set with the random forest model. I threw away all the columns where the other test set has only empty spaces, but there was also no extra information in these variables. Also the other models I tried (knn, rpart) give the same answers to the test set.