Activity tracking: Predicting exercise execution

Rona1d

august 6, 2016

```r
library(caret)
```

Introduction

This report was written in fulfillment of the assignment of week 4 of the 'Practical Machine Learning' course within the Coursera specialization track 'Data Science'.

Using devices such as *Jawbone Up*, *Nike FuelBand*, and *Fitbit* it is now possible to collect a large amount of data about personal activity relatively inexpensively. In this report data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants performing exercises is used. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways (more information is available here (see the section on the Weight Lifting Exercise Dataset)). Goal of this assignment is, by looking at the data, to predict in which way the exercises were performed.

Executive summary

Having analysed the data with machine learning techniques, we were able to accurately predict which type of exercise was performed. In particular the *Random Forest* model performed very well with an accuracy of 99.6%

Getting and cleaning data

For this research we will be using data on exercises provided through the Coursera course website. The data is devided in a training and test set.

```r
# First, make sure a folder named 'data' exists
if(!file.exists("data")) {
        dir.create("data")
}
setwd("./data")

# Downloading file...
fileURL1 <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
fileURL2 <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

if(!file.exists("HAT_traindata.csv")) {
        download.file(fileURL1, "./HAT_traindata.csv")
}
if(!file.exists("HAT_testdata.csv")) {
        download.file(fileURL2, "./HAT_testdata.csv")
}

# A quick preliminary scan of the data reveiled variables with a lot of blanks.
# It is therefore wise to name the blanks as 'NA' in the read.csv step
trainset <- read.csv("HAT_traindata.csv", na.strings=c(""," ","NA"))
valset <- read.csv("HAT_testdata.csv")
```

The data in the trainset (19622 rows and 160 columns) shows a lot of colums with a lot of missing values. To be specific, columns with more than 95% of NA's will be deleted from the dataset as they are of little use (please consult the Appendix for more detailed information about the variables). Note that the first 7 columns in the dataset are not providing much useful data either so those too wil be deleted.

```
trainset2 <- trainset[,colSums(is.na(trainset))<(nrow(trainset)*0.95)]
trainset3 <- subset(trainset2, select=-c(1:7)); rm(trainset2)
```

The final input dataset now has 'only' 53 columns left. To be able to perform cross validation on any model to be built, the dataset has to partitioned into a training set (80%) and a testing set (20%).

```
set.seed(1234)
inTrain = createDataPartition(trainset3$classe, p = 0.8)[[1]]
training = trainset3[ inTrain,]
testing = trainset3[-inTrain,]
```

Choosing and running models

The first choice for a model would be a *Random Forest* algorithm ("rf"). This type of model is very good for classification problems like the one in this report. It also, in general, outperforms (generalised) linear models when the relation between variables in a dataset is non-linear. For sake of comparison, a *Stochastic Gradient Boosting* ("gbm") model, which is another type of model based on decision trees, will also be fit on the dataset.

```
#fit1 <- train(classe~., method = "rf", data=training); save(fit1, file = "fit1.RData")
#fit2 <- train(classe~., method = "gbm", data=training); save(fit2, file = "fit2.RData")
#Note that the above fit will need to run when not run before.
#Once these fits are saved, one can pull the models from memory with the 'load' statement
load(file = "fit1.RData")
load(file = "fit2.RData")
```

Model performance

By predicting the fit models on the test dataset, the model performance can be established. Accuracy is measured by comparing the predicted values to the actual values in the test dataset.

```
p1 <- predict(fit1, training)
p2 <- predict(fit2, training)
p3 <- predict(fit1, testing)
p4 <- predict(fit2, testing)

c1 <- confusionMatrix(p1, training$classe)
c2 <- confusionMatrix(p2, training$classe)
c3 <- confusionMatrix(p3, testing$classe)
c4 <- confusionMatrix(p4, testing$classe)
```

On the training set, the rf model performs outstanding with an accuracy of 100%. The gbm model also has a high accuracy (97.4%) on the training set, but underperforms to the rf.The model to go with would therefore be the rf model. Performing cross-validation on the testset, the rf model still perfoms very good with aan accuracy of 99.6% which translates to an out of sample error rate of 0.4% (please refer to the Appendix for the model results). Combining both of the predictors would in general be a way to further increase accuracy, but with the current model accuracy of 99,6% that is not deemed necessary. The fitted rf model would therefore be the model of choice for predicting the performed exercises.

Appendix

```r
str(trainset) # Below an overview of variables in the initial dataset
```

```
## 'data.frame':    19622 obs. of  160 variables:
##  $ X                       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name               : Factor w/ 6 levels "adelmo","carlitos",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ raw_timestamp_part_1    : int  1323084231 1323084231 1323084231 1323084232 1323084232 1323084232 1
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 304277 368296 440390 484323 4844
##  $ cvtd_timestamp          : Factor w/ 20 levels "02/12/2011 13:32",..: 9 9 9 9 9 9 9 9 9 9 ...
##  $ new_window              : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ num_window              : int  11 11 11 12 12 12 12 12 12 12 ...
##  $ roll_belt               : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.45 ...
##  $ pitch_belt              : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.17 ...
##  $ yaw_belt                : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 ...
##  $ total_accel_belt        : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ kurtosis_roll_belt      : Factor w/ 396 levels "-0.016850","-0.021024",..: NA NA NA NA NA NA NA N
##  $ kurtosis_picth_belt     : Factor w/ 316 levels "-0.021887","-0.060755",..: NA NA NA NA NA NA NA N
##  $ kurtosis_yaw_belt       : Factor w/ 1 level "#DIV/0!": NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_roll_belt      : Factor w/ 394 levels "-0.003095","-0.010002",..: NA NA NA NA NA NA NA N
##  $ skewness_roll_belt.1    : Factor w/ 337 levels "-0.005928","-0.005960",..: NA NA NA NA NA NA NA N
##  $ skewness_yaw_belt       : Factor w/ 1 level "#DIV/0!": NA NA NA NA NA NA NA NA NA NA ...
##  $ max_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_belt          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_belt            : Factor w/ 67 levels "-0.1","-0.2",..: NA NA NA NA NA NA NA NA NA NA ...
##  $ min_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_belt          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_belt            : Factor w/ 67 levels "-0.1","-0.2",..: NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_roll_belt     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_belt    : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_belt      : Factor w/ 3 levels "#DIV/0!","0.00",..: NA NA NA NA NA NA NA NA NA NA .
##  $ var_total_accel_belt    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_belt        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_belt       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_belt         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ gyros_belt_x            : num  0 0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 0.03 ...
##  $ gyros_belt_y            : num  0 0 0 0 0.02 0 0 0 0 0 ...
##  $ gyros_belt_z            : num  -0.02 -0.02 -0.02 -0.03 -0.02 -0.02 -0.02 -0.02 -0.02 0 ...
##  $ accel_belt_x            : int  -21 -22 -20 -22 -21 -21 -22 -22 -20 -21 ...
##  $ accel_belt_y            : int  4 4 5 3 2 4 3 4 2 4 ...
##  $ accel_belt_z            : int  22 22 23 21 24 21 21 21 24 22 ...
##  $ magnet_belt_x           : int  -3 -7 -2 -6 -6 0 -4 -2 1 -3 ...
##  $ magnet_belt_y           : int  599 608 600 604 600 603 599 603 602 609 ...
##  $ magnet_belt_z           : int  -313 -311 -305 -310 -302 -312 -311 -313 -312 -308 ...
##  $ roll_arm                : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -128 ...
##  $ pitch_arm               : num  22.5 22.5 22.5 22.1 22.1 22 21.9 21.8 21.7 21.6 ...
##  $ yaw_arm                 : num  -161 -161 -161 -161 -161 -161 -161 -161 -161 -161 ...
##  $ total_accel_arm         : int  34 34 34 34 34 34 34 34 34 34 ...
##  $ var_accel_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_arm         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_arm        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_arm             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_arm          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_arm             : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ gyros_arm_x             : num  0 0.02 0.02 0.02 0 0.02 0 0.02 0.02 0.02 ...
##  $ gyros_arm_y             : num  0 -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -0.02 -0.03 -0.03 ...
##  $ gyros_arm_z             : num  -0.02 -0.02 -0.02 0.02 0 0 0 0 -0.02 -0.02 ...
##  $ accel_arm_x             : int  -288 -290 -289 -289 -289 -289 -289 -289 -288 -288 ...
##  $ accel_arm_y             : int  109 110 110 111 111 111 111 111 109 110 ...
##  $ accel_arm_z             : int  -123 -125 -126 -123 -123 -122 -125 -124 -122 -124 ...
##  $ magnet_arm_x            : int  -368 -369 -368 -372 -374 -369 -373 -372 -369 -376 ...
##  $ magnet_arm_y            : int  337 337 344 344 337 342 336 338 341 334 ...
##  $ magnet_arm_z            : int  516 513 513 512 506 513 509 510 518 516 ...
##  $ kurtosis_roll_arm       : Factor w/ 329 levels "-0.02438","-0.04190",..: NA NA NA NA NA NA NA NA
##  $ kurtosis_picth_arm      : Factor w/ 327 levels "-0.00484","-0.01311",..: NA NA NA NA NA NA NA NA
##  $ kurtosis_yaw_arm        : Factor w/ 394 levels "-0.01548","-0.01749",..: NA NA NA NA NA NA NA NA
##  $ skewness_roll_arm       : Factor w/ 330 levels "-0.00051","-0.00696",..: NA NA NA NA NA NA NA NA
##  $ skewness_pitch_arm      : Factor w/ 327 levels "-0.00184","-0.01185",..: NA NA NA NA NA NA NA NA
##  $ skewness_yaw_arm        : Factor w/ 394 levels "-0.00311","-0.00562",..: NA NA NA NA NA NA NA NA
##  $ max_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_arm             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_roll_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_arm             : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_roll_arm      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_arm     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_arm       : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ roll_dumbbell           : num  13.1 13.1 12.9 13.4 13.4 ...
##  $ pitch_dumbbell          : num  -70.5 -70.6 -70.3 -70.4 -70.4 ...
##  $ yaw_dumbbell            : num  -84.9 -84.7 -85.1 -84.9 -84.9 ...
##  $ kurtosis_roll_dumbbell  : Factor w/ 397 levels "-0.0035","-0.0073",..: NA NA NA NA NA NA NA NA NA
##  $ kurtosis_picth_dumbbell : Factor w/ 400 levels "-0.0163","-0.0233",..: NA NA NA NA NA NA NA NA NA
##  $ kurtosis_yaw_dumbbell   : Factor w/ 1 level "#DIV/0!": NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_roll_dumbbell  : Factor w/ 400 levels "-0.0082","-0.0096",..: NA NA NA NA NA NA NA NA NA
##  $ skewness_pitch_dumbbell : Factor w/ 401 levels "-0.0053","-0.0084",..: NA NA NA NA NA NA NA NA NA
##  $ skewness_yaw_dumbbell   : Factor w/ 1 level "#DIV/0!": NA NA NA NA NA NA NA NA NA NA ...
##  $ max_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_dumbbell        : Factor w/ 72 levels "-0.1","-0.2",..: NA NA NA NA NA NA NA NA NA NA ...
##  $ min_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_dumbbell        : Factor w/ 72 levels "-0.1","-0.2",..: NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_roll_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##   [list output truncated]
```

```r
c1$overall # Accuracy of the Random Forest model on the training set
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull 
##      1.0000000      1.0000000      0.9997651      1.0000000      0.2843493 
## AccuracyPValue  McnemarPValue 
##      0.0000000            NaN
```

```r
c2$overall # Accuracy of the Stochastic Gradient Boosting model on the training set
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull 
##   9.743933e-01   9.676052e-01   9.718009e-01   9.768078e-01   2.843493e-01 
## AccuracyPValue  McnemarPValue 
##   0.000000e+00   1.477962e-13
```

```r
c3 # Confusion Matrix and Statistics for the Random Forest model on the test set
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1116    5    0    0    0
##          B    0  752    3    0    0
##          C    0    2  680    3    0
##          D    0    0    1  640    1
##          E    0    0    0    0  720
## 
## Overall Statistics
##                                           
##                Accuracy : 0.9962          
##                  95% CI : (0.9937, 0.9979)
##     No Information Rate : 0.2845          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.9952          
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9908   0.9942   0.9953   0.9986
## Specificity            0.9982   0.9991   0.9985   0.9994   1.0000
## Pos Pred Value         0.9955   0.9960   0.9927   0.9969   1.0000
## Neg Pred Value         1.0000   0.9978   0.9988   0.9991   0.9997
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1917   0.1733   0.1631   0.1835
## Detection Prevalence   0.2858   0.1925   0.1746   0.1637   0.1835
## Balanced Accuracy      0.9991   0.9949   0.9963   0.9974   0.9993
```