# Peer-graded Assignment: Prediction Assignment Writeup

*D.R. Cirkel*

*2 maart 2018*

## Executive Summary

In this report, we try to fit a model in order to predict the class of the movement of a person wearing a Fitbit. We use three different technique. That is, random forest, regression tree and boosted forest. It is concluded that random forest has the highest accuracy. With the use of this model we predict the classes of the movements of people in a set without classes.

## Introduction

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. These type of devices are part of the quantified self movement - a group of enthusiasts who take measurements about themselves regularly to improve their health, to find patterns in their behavior, or because they are tech geeks. One thing that people regularly do is quantify how much of a particular activity they do, but they rarely quantify how well they do it. In this project, the goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har.

## Data

We load the data and have a quick look at the columns.

```
url.train <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
url.test <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
training.orig <- read.csv(url(url.train), na.strings = c("NA", "", "#DIV0!"))
testing.orig <- read.csv(url(url.test), na.strings = c("NA", "", "#DIV0!"))
```

When doing `head(training.orig)` there seem to be a lot of "NA"-columns. Also, the first 7 columns consist of meta data, which should not be related the classe column. We delete these columns as follows:

```
# remove first 7 columns
training <- training.orig[,-c(1:7)]
testing <- testing.orig[,-c(1:7)]

# remove all empty columns
training <- training[, colSums(is.na(training)) == 0]
testing <- testing[, colSums(is.na(testing)) == 0]
```

We found out there is no 'classe' column in the testing set. Therefore, we cannot test on that set. We have to divide our trainingset into our own train and test set.

```
library(caret)
```

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

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

```
inTrain <- createDataPartition(training$classe, p = 0.8, list = F)

training.new <- training[inTrain, ]
testing.new <- training[-inTrain, ]
```

## Models

We have chosen to fit a random forest (rf), regression tree (rpart) and boosted trees (gbm) for all the variables.

### Random forest

```
library(randomForest)
```

```
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
fit_rf <- randomForest(classe ~., data = training.new, method = "class")
pred_rf <- predict(fit_rf, newdata = testing.new)
confusionMatrix(testing.new$classe, pred_rf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1116    0    0    0    0
##          B    2  757    0    0    0
##          C    0    6  675    3    0
##          D    0    0    2  641    0
##          E    0    0    0    1  720
##
## Overall Statistics
##
##                Accuracy : 0.9964
##                  95% CI : (0.994, 0.998)
##     No Information Rate : 0.285
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9955
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9921   0.9970   0.9938   1.0000
## Specificity            1.0000   0.9994   0.9972   0.9994   0.9997
## Pos Pred Value         1.0000   0.9974   0.9868   0.9969   0.9986
```

```
## Neg Pred Value         0.9993   0.9981   0.9994   0.9988   1.0000
## Prevalence             0.2850   0.1945   0.1726   0.1644   0.1835
## Detection Rate         0.2845   0.1930   0.1721   0.1634   0.1835
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9991   0.9958   0.9971   0.9966   0.9998
```

We see this model has an accuracy of 99%, which is really high. However, we will still check the other models just to be sure.

### Regression tree

```
library(rpart)
fit_rpart <- rpart(classe ~., data = training.new, method = "class")
pred_rpart <- predict(fit_rpart, newdata = testing.new, type = "class")
confusionMatrix(testing.new$classe, pred_rpart)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1016   27   33   10   30
##          B  158  419   76   49   57
##          C   17   38  528   51   50
##          D   65   13  103  398   64
##          E   24   54   79   43  521
##
## Overall Statistics
##
##                Accuracy : 0.7346
##                  95% CI : (0.7205, 0.7484)
##     No Information Rate : 0.3263
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6627
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7937   0.7604   0.6447   0.7223   0.7216
## Specificity            0.9622   0.8992   0.9497   0.9273   0.9375
## Pos Pred Value         0.9104   0.5520   0.7719   0.6190   0.7226
## Neg Pred Value         0.9059   0.9583   0.9102   0.9534   0.9372
## Prevalence             0.3263   0.1405   0.2088   0.1405   0.1840
## Detection Rate         0.2590   0.1068   0.1346   0.1015   0.1328
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.8780   0.8298   0.7972   0.8248   0.8296
```

This model predicts with an accuracy of just 74%.

### Boosted trees

```
fit_gbm <- train(classe ~., data = training.new, method = "gbm", na.action=na.exclude, verbose=FALSE)
pred_gbm <- predict(fit_gbm, newdata = testing.new)
confusionMatrix(testing.new$classe, pred_gbm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1099   13    4    0    0
##          B   30  708   20    0    1
##          C    0   26  646   11    1
##          D    1    4   21  615    2
##          E    0    6    6   15  694
##
## Overall Statistics
##
##                Accuracy : 0.959
##                  95% CI : (0.9523, 0.965)
##     No Information Rate : 0.288
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9481
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9726   0.9353   0.9268   0.9594   0.9943
## Specificity            0.9939   0.9839   0.9882   0.9915   0.9916
## Pos Pred Value         0.9848   0.9328   0.9444   0.9565   0.9626
## Neg Pred Value         0.9890   0.9845   0.9843   0.9921   0.9988
## Prevalence             0.2880   0.1930   0.1777   0.1634   0.1779
## Detection Rate         0.2801   0.1805   0.1647   0.1568   0.1769
## Detection Prevalence   0.2845   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      0.9832   0.9596   0.9575   0.9755   0.9929
```

The accuracy of this model is 96%.

We can conclude that the random forest model predicts with the highest accuracy.

## Prediction

With this model, we can predict what the classe of the testing set will be:

```
predict(fit_rf, newdata = testing)
```

```
##  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
##  B  A  B  A  A  E  D  B  A  A  B  C  B  A  E  E  A  B  B  B
## Levels: A B C D E
```