# Practical Machine learning

*Joachim Allouche*

*7 apr 2019*

## Practical Machine Learning Final Project

This project aim to predict how people did a certain exercise, using data fromuse data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. The data was collected by Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013.

The paper contains a descibtion off how the data was prepared picking the relevant variables and splitting it in to a training ad testing set. After that there will be describtion and evaluation off Cross Validation and the two methods I have tried, Classification Tree and Random Forest. Finally I conclude that Random Forest is the best method for predickting how people did the exercise. To make the text readable some of the code has been hidden, at the end of the paper you will find all the code.

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

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

### Loading and preparing the data

The data is loaded from the same folder as my script.

```
trainingdata <- read.csv("pml-training.csv", header = TRUE)
testdata <- read.csv("pml-testing.csv",header = TRUE)
```

### Picking Features (Removing Variables)

A lot of the variables in the data-set only contains NA's (See Appendix 1) these are removed in order to simplify the data-set.

```
trainingdata <- trainingdata[,colSums(is.na(trainingdata))=="0"]
testdata <- testdata[,colSums(is.na(testdata))=="0"]
```

After that i am checking for variables with non to littel variability The variables that have non to little variability are then removed.

```
nzv <- nearZeroVar(trainingdata)
trainingdata <- trainingdata[,-nzv]
nzv <- nearZeroVar(testdata)
testdata <- testdata[,-nzv]
```

Finally the first 6 variables are removed as they do not have any impact on the outcome. This leave us with a trainingset containing 19622 observations and 53 variables.

```
trainingdata <- trainingdata[,-c(1:6)]
testdata <- testdata[,-c(1:6)]

dim(trainingdata)
```

```
## [1] 19622    53
```

### slicing the data

In order to test different model without risking overfitting, the trainingset is split in to a smaller training set and a test set.

```
set.seed(534)
intrain <- createDataPartition(trainingdata$classe, p = 0.7, list = FALSE)
traindata <- trainingdata[intrain,]
validdata <- trainingdata[-intrain,]
```

### Cross Validation Classification Tree and Random forest

To predict the outcome I am using to different methods first I am preparing my Cross Validation code which I will be using in order to limit the effect of overfitting. The function that i am using is the trainControl function with 5 folds by the end i am using ConfusionMatix to make the results clearer.

```
crva <- trainControl(method = "cv", number=5)
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

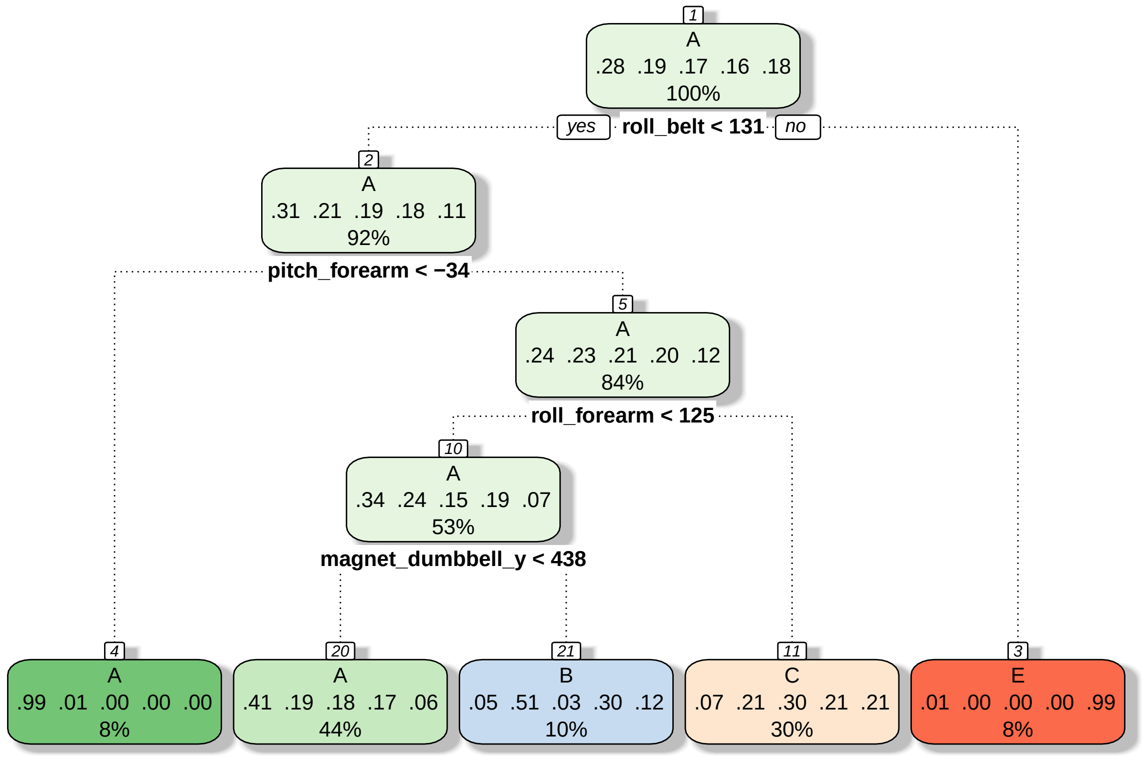
```
##
## Attaching package: 'rattle'
```

```
## The following object is masked from 'package:randomForest':
##
##     importance
```

### Classification Tree

My first method is classification tree the result clearly an accuracy off .482 clearly indicates that this method is not productive.

```
model1 <- train(classe~.,data = traindata, method="rpart", trControl=crva)
fancyRpartPlot(model1$finalModel)
```



Rattle 2019−apr−07 22:41:20 Joachim

```
p1 <- predict(model1, newdata = validdata)
confm1 <- confusionMatrix(validdata$classe,p1)
confm1$overall[1]
```

```
##  Accuracy
## 0.4820731
```

### Random Forest Model

The next method is Random Forest This method is giving us an accuracy off .9947 and an out sample error of .00053, which is better than the Accuracy rate of the Classification Tree and in

```
model2 <- randomForest(classe~.,data = traindata, method="rf", trControl=crva, verbose=FALSE)
p2 <- predict(model2, newdata = validdata)
```

```
confm2 <- confusionMatrix(validdata$classe,p2)
confm2
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1672    1    0    0    1
##          B    5 1134    0    0    0
##          C    0   11 1015    0    0
##          D    0    0    9  955    0
##          E    0    0    0    4 1078
##
## Overall Statistics
##
##                Accuracy : 0.9947
##                  95% CI : (0.9925, 0.9964)
##     No Information Rate : 0.285
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9933
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9970   0.9895   0.9912   0.9958   0.9991
## Specificity            0.9995   0.9989   0.9977   0.9982   0.9992
## Pos Pred Value         0.9988   0.9956   0.9893   0.9907   0.9963
## Neg Pred Value         0.9988   0.9975   0.9981   0.9992   0.9998
## Prevalence             0.2850   0.1947   0.1740   0.1630   0.1833
## Detection Rate         0.2841   0.1927   0.1725   0.1623   0.1832
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9983   0.9942   0.9945   0.9970   0.9991
```

### Conclusion Accuracy and Out of Sample Error

The test clearly indicate that Random Forest is the best fitt, therefore the model with Random Forest will be used in the prediction Quiz.

The Accuracy is .9947 The Out of Sample Error is .00053

```
predict(model2,newdata = testdata)
```

```
##  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
```

## Appendix 1

```
str(trainingdata)
```

```
## 'data.frame':    19622 obs. of  53 variables:
##  $ 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 ...
##  $ 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 ...
##  $ 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 ...
##  $ 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 ...
##  $ total_accel_dumbbell: int  37 37 37 37 37 37 37 37 37 37 ...
##  $ gyros_dumbbell_x    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ gyros_dumbbell_y    : num  -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 ...
##  $ gyros_dumbbell_z    : num  0 0 0 -0.02 0 0 0 0 0 0 ...
##  $ accel_dumbbell_x    : int  -234 -233 -232 -232 -233 -234 -232 -234 -232 -235 ...
##  $ accel_dumbbell_y    : int  47 47 46 48 48 48 47 46 47 48 ...
##  $ accel_dumbbell_z    : int  -271 -269 -270 -269 -270 -269 -270 -272 -269 -270 ...
##  $ magnet_dumbbell_x   : int  -559 -555 -561 -552 -554 -558 -551 -555 -549 -558 ...
##  $ magnet_dumbbell_y   : int  293 296 298 303 292 294 295 300 292 291 ...
##  $ magnet_dumbbell_z   : num  -65 -64 -63 -60 -68 -66 -70 -74 -65 -69 ...
##  $ roll_forearm        : num  28.4 28.3 28.3 28.1 28 27.9 27.9 27.8 27.7 27.7 ...
##  $ pitch_forearm       : num  -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.8 -63.8 -63.8 ...
##  $ yaw_forearm         : num  -153 -153 -152 -152 -152 -152 -152 -152 -152 -152 ...
##  $ total_accel_forearm : int  36 36 36 36 36 36 36 36 36 36 ...
##  $ gyros_forearm_x     : num  0.03 0.02 0.03 0.02 0.02 0.02 0.02 0.02 0.03 0.02 ...
##  $ gyros_forearm_y     : num  0 0 -0.02 -0.02 0 -0.02 0 -0.02 0 0 ...
##  $ gyros_forearm_z     : num  -0.02 -0.02 0 0 -0.02 -0.03 -0.02 0 -0.02 -0.02 ...
##  $ accel_forearm_x     : int  192 192 196 189 189 193 195 193 193 190 ...
##  $ accel_forearm_y     : int  203 203 204 206 206 203 205 205 204 205 ...
##  $ accel_forearm_z     : int  -215 -216 -213 -214 -214 -215 -215 -213 -214 -215 ...
##  $ magnet_forearm_x    : int  -17 -18 -18 -16 -17 -9 -18 -9 -16 -22 ...
##  $ magnet_forearm_y    : num  654 661 658 658 655 660 659 660 653 656 ...
##  $ magnet_forearm_z    : num  476 473 469 469 473 478 470 474 476 473 ...
##  $ classe              : Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...
```

## Appendix 2: code

```r
library(ggplot2)
library(caret)
library(randomForest)
trainingdata <- read.csv("pml-training.csv", header = TRUE)
testdata <- read.csv("pml-testing.csv",header = TRUE)
 str(trainingdata)
 trainingdata <- trainingdata[,colSums(is.na(trainingdata))=="0"]
testdata <- testdata[,colSums(is.na(testdata))=="0"]
trainingdata <- trainingdata[,-c(1:6)]
testdata <- testdata[,-c(1:6)]
dim(trainingdata)
set.seed(534)
intrain <- createDataPartition(trainingdata$classe, p = 0.7, list = FALSE)
traindata <- trainingdata[intrain,]
validdata <- trainingdata[-intrain,]
crva <- trainControl(method = "cv", number=5)
library(rattle)
suppressMessages(library(rattle))
model1 <- train(classe~.,data = traindata, method="rpart", trControl=crva)
fancyRpartPlot(model1$finalModel)
p1 <- predict(model1, newdata = validdata)
confm1 <- confusionMatrix(validdata$classe,p1)
confm1$overall[1]
model2 <- randomForest(classe~.,data = traindata, method="rf", trControl=crva, verbose=FALSE)
p2 <- predict(model2, newdata = validdata)
confm2 <- confusionMatrix(validdata$classe,p2)
confm2
predict(model2,newdata = testdata)
```