

# Practical Machine Learning Project - Quantified Self Movement Data Analysis Report

*Bogdan Oancea*

## 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, we will use data from accelerometers on the belt, forearm, arm, and dumbbell of 6 participants to predict the manner in which they did the exercise.

## First step: Data Preprocessing

```
library(caret)
```

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

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

```
## randomForest 4.6-10  
## Type rfNews() to see new features/changes/bug fixes.
```

```
library(corrplot)
```

## A. Download the Datasets

```
trainUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"  
testUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"  
trainFile <- "./datasets/pml-training.csv"  
testFile <- "./datasets/pml-testing.csv"  
if (!file.exists("./datasets")) {  
  dir.create("./datasets")  
}  
if (!file.exists(trainFile)) {  
  download.file(trainUrl, destfile=trainFile, method="curl")  
}  
if (!file.exists(testFile)) {  
  download.file(testUrl, destfile=testFile, method="curl")  
}
```

## B. Read the Datasets into dataframe objects

We can read now the two .csv files into two dataframe objects.

```
trainRaw <- read.csv("./datasets/pml-training.csv")
testRaw <- read.csv("./datasets/pml-testing.csv")
dim(trainRaw)
```

```
## [1] 19622 160
```

```
dim(testRaw)
```

```
## [1] 20 160
```

The training dataset contains 19622 observations and 160 variables, while the testing dataset contains 20 observations and 160 variables. The “classe” variable in the training set is the outcome to predict.

## C. Clean the data

Clean the dataset and get rid of observations with missing values as well as some meaningless variables.

```
sum(complete.cases(trainRaw))
```

```
## [1] 406
```

Remove the columns that contain missing values (NA).

```
trainRaw <- trainRaw[, colSums(is.na(trainRaw)) == 0]
testRaw <- testRaw[, colSums(is.na(testRaw)) == 0]
```

Get rid of some columns that do not contribute much to the accelerometer measurements.

```
classe <- trainRaw$classe
trainRemove <- grepl("^X|timestamp|window", names(trainRaw))
trainRaw <- trainRaw[, !trainRemove]
trainCleaned <- trainRaw[, sapply(trainRaw, is.numeric)]
trainCleaned$classe <- classe
testRemove <- grepl("^X|timestamp|window", names(testRaw))
testRaw <- testRaw[, !testRemove]
testCleaned <- testRaw[, sapply(testRaw, is.numeric)]
```

The cleaned training dataset contains 19622 observations and 53 variables, while the testing dataset contains 20 observations and 53 variables. The “classe” variable is still in the cleaned training set.

## D. Slice the data

We can split the cleaned training set into a pure training data set (70%) and a validation data set (30%). We will use the validation dataset for cross validation in future steps.

```
set.seed(22519) # For reproducible purpose
inTrain <- createDataPartition(trainCleaned$classe, p=0.70, list=F)
trainData <- trainCleaned[inTrain, ]
testData <- trainCleaned[-inTrain, ]
```

## Second Step: Data Modeling

We fit a predictive model for activity recognition using **Random Forest** algorithm because it automatically selects important variables and is robust to correlated covariates & outliers in general. We will use **5-fold cross validation** when applying the algorithm.

```
controlRf <- trainControl(method="cv", 5)
modelRf <- train(classe ~ ., data=trainData, method="rf", trControl=controlRf, ntree=250)
modelRf
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##    5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
##
## Summary of sample sizes: 10989, 10989, 10991, 10990, 10989
##
## Resampling results across tuning parameters:
##
##  mtry  Accuracy   Kappa      Accuracy SD   Kappa SD
##    2    0.9910462 0.9886729 0.001301269   0.001647776
##   27    0.9914102 0.9891334 0.001717547   0.002174708
##   52    0.9850037 0.9810264 0.002718384   0.003439965
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 27.
```

We estimate the performance of the model on the validation dataset.

```
predictRf <- predict(modelRf, testData)
confusionMatrix(testData$classe, predictRf)
```

```
## Confusion Matrix and Statistics
##
##              Reference
## Prediction    A    B    C    D    E
##      A 1672     1     0     0     1
##      B     7 1128     4     0     0
##      C     0     0 1021     5     0
##      D     0     0   14  949     1
##      E     0     0    1   7 1074
##
```

```
## Overall Statistics
##
##           Accuracy : 0.993
##           95% CI : (0.9906, 0.995)
##       No Information Rate : 0.2853
##       P-Value [Acc > NIR] : < 2.2e-16
##
##           Kappa : 0.9912
##  McNemar's Test P-Value : NA
##
## Statistics by Class:
##
##           Class: A Class: B Class: C Class: D Class: E
## Sensitivity      0.9958  0.9991  0.9817  0.9875  0.9981
## Specificity      0.9995  0.9977  0.9990  0.9970  0.9983
## Pos Pred Value   0.9988  0.9903  0.9951  0.9844  0.9926
## Neg Pred Value   0.9983  0.9998  0.9961  0.9976  0.9996
## Prevalence       0.2853  0.1918  0.1767  0.1633  0.1828
## Detection Rate   0.2841  0.1917  0.1735  0.1613  0.1825
## Detection Prevalence 0.2845  0.1935  0.1743  0.1638  0.1839
## Balanced Accuracy 0.9977  0.9984  0.9903  0.9922  0.9982
```

```
accuracy <- postResample(predictRf, testData$classe)
accuracy
```

```
## Accuracy      Kappa
## 0.9930331 0.9911872
```

```
oos <- 1 - as.numeric(confusionMatrix(testData$classe, predictRf)$overall[1])
oos
```

```
## [1] 0.006966865
```

The estimated accuracy of the model is 99.31% and the estimated out-of-sample error is 0.69%.

### Third step: Predicting for Test Dataset

We apply the model to the original testing dataset. We remove the `problem_id` column first.

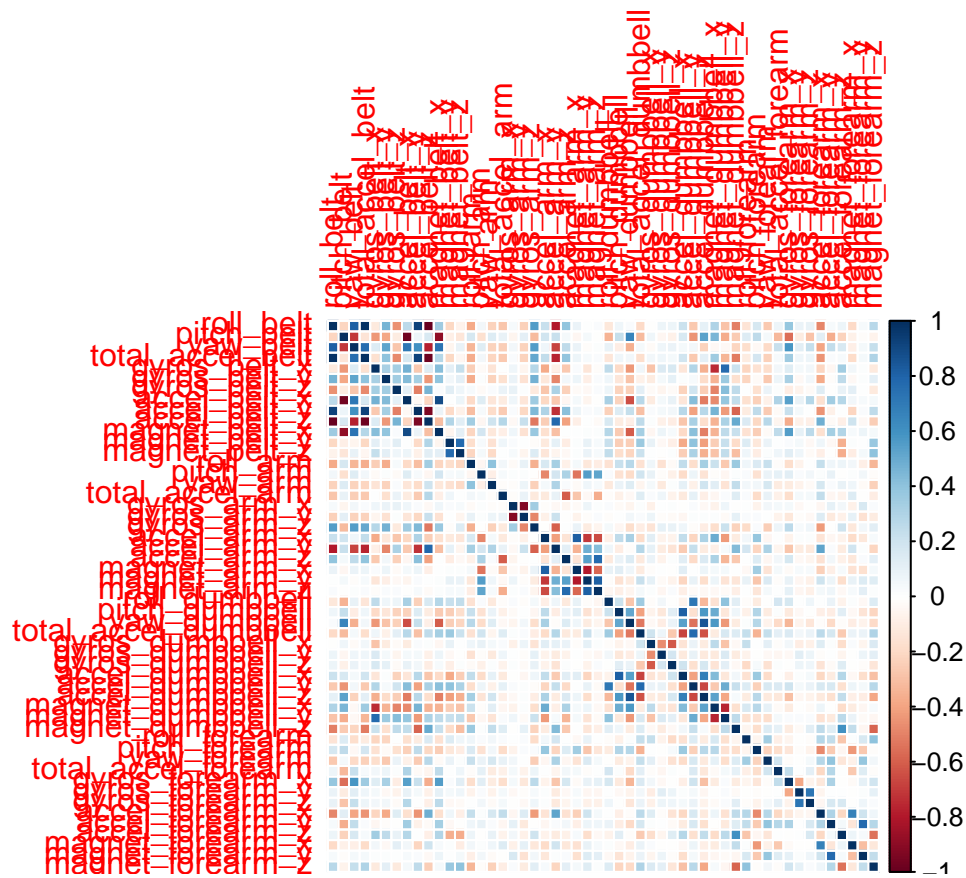
```
result <- predict(modelRf, testCleaned[, -length(names(testCleaned))])
result
```

```
## [1] 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: Figures

### 1. Correlation Matrix Visualization

```
corrPlot <- cor(trainData[, -length(names(trainData))])
corrplot(corrPlot, method="color")
```



## 2. Decision Tree Visualization

```
treeModel <- rpart(classe ~ ., data=trainData, method="class")
prp(treeModel) # fast plot
```

