Coursera Practical Machine Learning

Thomas H

Saturday, December 20, 2014

Background to the Project

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, your 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. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

Data for the Project

The training data for this project are available at https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv).

The test data are available at https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv).

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har).

Goals of the Project

The goal of this project is to predict the manner in which they did the exercise. This is the “classe” variable in the training set. Other variables are used to predict “classe”. This report describes how to built the prediction model, how cross validation was used, what the expected out of sample error is, and why the choices were maid. The prediction model was used to predict 20 different test cases.

R code for prediction

In the working directory load libraries, read the data, delete empty or almost empty columns and identifying columns that cannot explain the response (such as time).

```
library(caret)
```

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

```
library(kernlab)
library(rpart)

# SETTING THE SEED TO GUARANTEE THE REPRODUCIBILITY.
set.seed(1985)

# Training dataset:
training.data <- read.csv("pml-training.csv",header=TRUE,na.strings=c("NA","#DIV/0!", ""))
str(training.data)
```

```
## '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 132308
4232 1323084232 1323084232 1323084232 1323084232 ...
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 304277 368296 440390 48432
3 484434 ...
##  $ 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      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_picth_belt     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_yaw_belt       : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_belt      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_roll_belt.1    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_yaw_belt       : logi  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            : num  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            : num  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      : num  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       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_picth_arm      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_yaw_arm        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_roll_arm       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_pitch_arm      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_yaw_arm        : num  NA NA 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  : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_picth_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_yaw_dumbbell   : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_dumbbell  : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_pitch_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_yaw_dumbbell   : logi  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        : num  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        : num  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]
```

```
#head(training.data)
#summary(training.data)

# Delete missing values (100 variables will be deleted):
training.data <- training.data[,colSums(is.na(training.data)) == 0]
# Delete not required colums
not_required <- "avg|stddev|var|min|max|amplitude|skewness|kurtosis|timestamp|user_name|new_window
"
throwcols<-grep(not_required,names(training.data),value=FALSE)
training.data<-training.data[,-throwcols]
str(training.data)
```

```
## 'data.frame':    19622 obs. of  55 variables:
##  $ X                   : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ 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 ...
##  $ 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 ...
```

```
# Testing dataset:
testing.data <- read.csv("pml-testing.csv",header=TRUE,na.strings=c("NA","#DIV/0!", ""))
str(testing.data)
```

```
## 'data.frame':    20 obs. of  160 variables:
##  $ X                       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name               : Factor w/ 6 levels "adelmo","carlitos",..: 6 5 5 1 4 5 5 5 2 3 ...
##  $ raw_timestamp_part_1    : int  1323095002 1322673067 1322673075 1322832789 1322489635 132267
3149 1322673128 1322673076 1323084240 1322837822 ...
##  $ raw_timestamp_part_2    : int  868349 778725 342967 560311 814776 510661 766645 54671 916313
 384285 ...
##  $ cvtd_timestamp          : Factor w/ 11 levels "02/12/2011 13:33",..: 5 10 10 1 6 11 11 10 3
2 ...
```

```
##  $ new_window             : Factor w/ 1 level "no": 1 1 1 1 1 1 1 1 1 1 ...
##  $ num_window             : int  74 431 439 194 235 504 485 440 323 664 ...
##  $ roll_belt              : num  123 1.02 0.87 125 1.35 -5.92 1.2 0.43 0.93 114 ...
##  $ pitch_belt             : num  27 4.87 1.82 -41.6 3.33 1.59 4.44 4.15 6.72 22.4 ...
##  $ yaw_belt               : num  -4.75 -88.9 -88.5 162 -88.6 -87.7 -87.3 -88.5 -93.7 -13.1 ...
##  $ total_accel_belt       : int  20 4 5 17 3 4 4 4 4 18 ...
##  $ kurtosis_roll_belt     : logi  NA NA NA NA NA NA ...
##  $ kurtosis_picth_belt    : logi  NA NA NA NA NA NA ...
##  $ kurtosis_yaw_belt      : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_belt     : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_belt.1   : logi  NA NA NA NA NA NA ...
##  $ skewness_yaw_belt      : logi  NA NA NA NA NA NA ...
##  $ max_roll_belt          : logi  NA NA NA NA NA NA ...
##  $ max_picth_belt         : logi  NA NA NA NA NA NA ...
##  $ max_yaw_belt           : logi  NA NA NA NA NA NA ...
##  $ min_roll_belt          : logi  NA NA NA NA NA NA ...
##  $ min_pitch_belt         : logi  NA NA NA NA NA NA ...
##  $ min_yaw_belt           : logi  NA NA NA NA NA NA ...
##  $ amplitude_roll_belt    : logi  NA NA NA NA NA NA ...
##  $ amplitude_pitch_belt   : logi  NA NA NA NA NA NA ...
##  $ amplitude_yaw_belt     : logi  NA NA NA NA NA NA ...
##  $ var_total_accel_belt   : logi  NA NA NA NA NA NA ...
##  $ avg_roll_belt          : logi  NA NA NA NA NA NA ...
##  $ stddev_roll_belt       : logi  NA NA NA NA NA NA ...
##  $ var_roll_belt          : logi  NA NA NA NA NA NA ...
##  $ avg_pitch_belt         : logi  NA NA NA NA NA NA ...
##  $ stddev_pitch_belt      : logi  NA NA NA NA NA NA ...
##  $ var_pitch_belt         : logi  NA NA NA NA NA NA ...
##  $ avg_yaw_belt           : logi  NA NA NA NA NA NA ...
##  $ stddev_yaw_belt        : logi  NA NA NA NA NA NA ...
##  $ var_yaw_belt           : logi  NA NA NA NA NA NA ...
##  $ gyros_belt_x           : num  -0.5 -0.06 0.05 0.11 0.03 0.1 -0.06 -0.18 0.1 0.14 ...
##  $ gyros_belt_y           : num  -0.02 -0.02 0.02 0.11 0.02 0.05 0 -0.02 0 0.11 ...
##  $ gyros_belt_z           : num  -0.46 -0.07 0.03 -0.16 0 -0.13 0 -0.03 -0.02 -0.16 ...
##  $ accel_belt_x           : int  -38 -13 1 46 -8 -11 -14 -10 -15 -25 ...
##  $ accel_belt_y           : int  69 11 -1 45 4 -16 2 -2 1 63 ...
##  $ accel_belt_z           : int  -179 39 49 -156 27 38 35 42 32 -158 ...
##  $ magnet_belt_x          : int  -13 43 29 169 33 31 50 39 -6 10 ...
##  $ magnet_belt_y          : int  581 636 631 608 566 638 622 635 600 601 ...
##  $ magnet_belt_z          : int  -382 -309 -312 -304 -418 -291 -315 -305 -302 -330 ...
##  $ roll_arm               : num  40.7 0 0 -109 76.1 0 0 0 -137 -82.4 ...
##  $ pitch_arm              : num  -27.8 0 0 55 2.76 0 0 0 11.2 -63.8 ...
##  $ yaw_arm                : num  178 0 0 -142 102 0 0 0 -167 -75.3 ...
##  $ total_accel_arm        : int  10 38 44 25 29 14 15 22 34 32 ...
##  $ var_accel_arm          : logi  NA NA NA NA NA NA ...
##  $ avg_roll_arm           : logi  NA NA NA NA NA NA ...
##  $ stddev_roll_arm        : logi  NA NA NA NA NA NA ...
##  $ var_roll_arm           : logi  NA NA NA NA NA NA ...
##  $ avg_pitch_arm          : logi  NA NA NA NA NA NA ...
##  $ stddev_pitch_arm       : logi  NA NA NA NA NA NA ...
```

```
##  $ var_pitch_arm           : logi  NA NA NA NA NA NA ...
##  $ avg_yaw_arm             : logi  NA NA NA NA NA NA ...
##  $ stddev_yaw_arm          : logi  NA NA NA NA NA NA ...
##  $ var_yaw_arm             : logi  NA NA NA NA NA NA ...
##  $ gyros_arm_x             : num  -1.65 -1.17 2.1 0.22 -1.96 0.02 2.36 -3.71 0.03 0.26 ...
##  $ gyros_arm_y             : num  0.48 0.85 -1.36 -0.51 0.79 0.05 -1.01 1.85 -0.02 -0.5 ...
##  $ gyros_arm_z             : num  -0.18 -0.43 1.13 0.92 -0.54 -0.07 0.89 -0.69 -0.02 0.79 ...
##  $ accel_arm_x             : int  16 -290 -341 -238 -197 -26 99 -98 -287 -301 ...
##  $ accel_arm_y             : int  38 215 245 -57 200 130 79 175 111 -42 ...
##  $ accel_arm_z             : int  93 -90 -87 6 -30 -19 -67 -78 -122 -80 ...
##  $ magnet_arm_x            : int  -326 -325 -264 -173 -170 396 702 535 -367 -420 ...
##  $ magnet_arm_y            : int  385 447 474 257 275 176 15 215 335 294 ...
##  $ magnet_arm_z            : int  481 434 413 633 617 516 217 385 520 493 ...
##  $ kurtosis_roll_arm       : logi  NA NA NA NA NA NA ...
##  $ kurtosis_picth_arm      : logi  NA NA NA NA NA NA ...
##  $ kurtosis_yaw_arm        : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_arm       : logi  NA NA NA NA NA NA ...
##  $ skewness_pitch_arm      : logi  NA NA NA NA NA NA ...
##  $ skewness_yaw_arm        : logi  NA NA NA NA NA NA ...
##  $ max_roll_arm            : logi  NA NA NA NA NA NA ...
##  $ max_picth_arm           : logi  NA NA NA NA NA NA ...
##  $ max_yaw_arm             : logi  NA NA NA NA NA NA ...
##  $ min_roll_arm            : logi  NA NA NA NA NA NA ...
##  $ min_pitch_arm           : logi  NA NA NA NA NA NA ...
##  $ min_yaw_arm             : logi  NA NA NA NA NA NA ...
##  $ amplitude_roll_arm      : logi  NA NA NA NA NA NA ...
##  $ amplitude_pitch_arm     : logi  NA NA NA NA NA NA ...
##  $ amplitude_yaw_arm       : logi  NA NA NA NA NA NA ...
##  $ roll_dumbbell           : num  -17.7 54.5 57.1 43.1 -101.4 ...
##  $ pitch_dumbbell          : num  25 -53.7 -51.4 -30 -53.4 ...
##  $ yaw_dumbbell            : num  126.2 -75.5 -75.2 -103.3 -14.2 ...
##  $ kurtosis_roll_dumbbell  : logi  NA NA NA NA NA NA ...
##  $ kurtosis_picth_dumbbell : logi  NA NA NA NA NA NA ...
##  $ kurtosis_yaw_dumbbell   : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_dumbbell  : logi  NA NA NA NA NA NA ...
##  $ skewness_pitch_dumbbell : logi  NA NA NA NA NA NA ...
##  $ skewness_yaw_dumbbell   : logi  NA NA NA NA NA NA ...
##  $ max_roll_dumbbell       : logi  NA NA NA NA NA NA ...
##  $ max_picth_dumbbell      : logi  NA NA NA NA NA NA ...
##  $ max_yaw_dumbbell        : logi  NA NA NA NA NA NA ...
##  $ min_roll_dumbbell       : logi  NA NA NA NA NA NA ...
##  $ min_pitch_dumbbell      : logi  NA NA NA NA NA NA ...
##  $ min_yaw_dumbbell        : logi  NA NA NA NA NA NA ...
##  $ amplitude_roll_dumbbell : logi  NA NA NA NA NA NA ...
##   [list output truncated]
```

```
#head(testing.data)
#summary(testing.data)

# Delete missing values (100 variables will be deleted):
testing.data <- testing.data[,colSums(is.na(testing.data)) == 0]
# Delete not required colums
not_required <- "avg|stddev|var|min|max|amplitude|skewness|kurtosis|timestamp|user_name|new_window
"
throwcols<-grep(not_required,names(testing.data),value=FALSE)
testing.data<-testing.data[,-throwcols]
str(testing.data)
```

```
## 'data.frame':    20 obs. of  55 variables:
##  $ X                    : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ num_window           : int  74 431 439 194 235 504 485 440 323 664 ...
##  $ roll_belt            : num  123 1.02 0.87 125 1.35 -5.92 1.2 0.43 0.93 114 ...
##  $ pitch_belt           : num  27 4.87 1.82 -41.6 3.33 1.59 4.44 4.15 6.72 22.4 ...
##  $ yaw_belt             : num  -4.75 -88.9 -88.5 162 -88.6 -87.7 -87.3 -88.5 -93.7 -13.1 ...
##  $ total_accel_belt     : int  20 4 5 17 3 4 4 4 4 18 ...
##  $ gyros_belt_x         : num  -0.5 -0.06 0.05 0.11 0.03 0.1 -0.06 -0.18 0.1 0.14 ...
##  $ gyros_belt_y         : num  -0.02 -0.02 0.02 0.11 0.02 0.05 0 -0.02 0 0.11 ...
##  $ gyros_belt_z         : num  -0.46 -0.07 0.03 -0.16 0 -0.13 0 -0.03 -0.02 -0.16 ...
##  $ accel_belt_x         : int  -38 -13 1 46 -8 -11 -14 -10 -15 -25 ...
##  $ accel_belt_y         : int  69 11 -1 45 4 -16 2 -2 1 63 ...
##  $ accel_belt_z         : int  -179 39 49 -156 27 38 35 42 32 -158 ...
##  $ magnet_belt_x        : int  -13 43 29 169 33 31 50 39 -6 10 ...
##  $ magnet_belt_y        : int  581 636 631 608 566 638 622 635 600 601 ...
##  $ magnet_belt_z        : int  -382 -309 -312 -304 -418 -291 -315 -305 -302 -330 ...
##  $ roll_arm             : num  40.7 0 0 -109 76.1 0 0 0 -137 -82.4 ...
##  $ pitch_arm            : num  -27.8 0 0 55 2.76 0 0 0 11.2 -63.8 ...
##  $ yaw_arm              : num  178 0 0 -142 102 0 0 0 -167 -75.3 ...
##  $ total_accel_arm      : int  10 38 44 25 29 14 15 22 34 32 ...
##  $ gyros_arm_x          : num  -1.65 -1.17 2.1 0.22 -1.96 0.02 2.36 -3.71 0.03 0.26 ...
##  $ gyros_arm_y          : num  0.48 0.85 -1.36 -0.51 0.79 0.05 -1.01 1.85 -0.02 -0.5 ...
##  $ gyros_arm_z          : num  -0.18 -0.43 1.13 0.92 -0.54 -0.07 0.89 -0.69 -0.02 0.79 ...
##  $ accel_arm_x          : int  16 -290 -341 -238 -197 -26 99 -98 -287 -301 ...
##  $ accel_arm_y          : int  38 215 245 -57 200 130 79 175 111 -42 ...
##  $ accel_arm_z          : int  93 -90 -87 6 -30 -19 -67 -78 -122 -80 ...
##  $ magnet_arm_x         : int  -326 -325 -264 -173 -170 396 702 535 -367 -420 ...
##  $ magnet_arm_y         : int  385 447 474 257 275 176 15 215 335 294 ...
##  $ magnet_arm_z         : int  481 434 413 633 617 516 217 385 520 493 ...
##  $ roll_dumbbell        : num  -17.7 54.5 57.1 43.1 -101.4 ...
##  $ pitch_dumbbell       : num  25 -53.7 -51.4 -30 -53.4 ...
##  $ yaw_dumbbell         : num  126.2 -75.5 -75.2 -103.3 -14.2 ...
##  $ total_accel_dumbbell: int  9 31 29 18 4 29 29 29 3 2 ...
##  $ gyros_dumbbell_x     : num  0.64 0.34 0.39 0.1 0.29 -0.59 0.34 0.37 0.03 0.42 ...
##  $ gyros_dumbbell_y     : num  0.06 0.05 0.14 -0.02 -0.47 0.8 0.16 0.14 -0.21 0.51 ...
##  $ gyros_dumbbell_z     : num  -0.61 -0.71 -0.34 0.05 -0.46 1.1 -0.23 -0.39 -0.21 -0.03 ...
##  $ accel_dumbbell_x     : int  21 -153 -141 -51 -18 -138 -145 -140 0 -7 ...
```

```
##  $ accel_dumbbell_y    : int  -15 155 155 72 -30 166 150 159 25 -20 ...
##  $ accel_dumbbell_z    : int  81 -205 -196 -148 -5 -186 -190 -191 9 7 ...
##  $ magnet_dumbbell_x   : int  523 -502 -506 -576 -424 -543 -484 -515 -519 -531 ...
##  $ magnet_dumbbell_y   : int  -528 388 349 238 252 262 354 350 348 321 ...
##  $ magnet_dumbbell_z   : int  -56 -36 41 53 312 96 97 53 -32 -164 ...
##  $ roll_forearm        : num  141 109 131 0 -176 150 155 -161 15.5 13.2 ...
##  $ pitch_forearm       : num  49.3 -17.6 -32.6 0 -2.16 1.46 34.5 43.6 -63.5 19.4 ...
##  $ yaw_forearm         : num  156 106 93 0 -47.9 89.7 152 -89.5 -139 -105 ...
##  $ total_accel_forearm : int  33 39 34 43 24 43 32 47 36 24 ...
##  $ gyros_forearm_x     : num  0.74 1.12 0.18 1.38 -0.75 -0.88 -0.53 0.63 0.03 0.02 ...
##  $ gyros_forearm_y     : num  -3.34 -2.78 -0.79 0.69 3.1 4.26 1.8 -0.74 0.02 0.13 ...
##  $ gyros_forearm_z     : num  -0.59 -0.18 0.28 1.8 0.8 1.35 0.75 0.49 -0.02 -0.07 ...
##  $ accel_forearm_x     : int  -110 212 154 -92 131 230 -192 -151 195 -212 ...
##  $ accel_forearm_y     : int  267 297 271 406 -93 322 170 -331 204 98 ...
##  $ accel_forearm_z     : int  -149 -118 -129 -39 172 -144 -175 -282 -217 -7 ...
##  $ magnet_forearm_x    : int  -714 -237 -51 -233 375 -300 -678 -109 0 -403 ...
##  $ magnet_forearm_y    : int  419 791 698 783 -787 800 284 -619 652 723 ...
##  $ magnet_forearm_z    : int  617 873 783 521 91 884 585 -32 469 512 ...
##  $ problem_id          : int  1 2 3 4 5 6 7 8 9 10 ...
```

```
# Drop classe colum
training.data[,1]<-NULL
testing.data[,1]<-NULL
```

Now that we are working with clean data, subset the training into train and test. 75% of the training data is used to build the model, and 25% is used to validate the model and to estimate the out of sample error.

```
library(caret)
inTrain <- createDataPartition(y=training.data$classe,p=0.75,list=FALSE)
training.train <- training.data[inTrain,]
training.test <- training.data[-inTrain,]
dim(training.train); dim(training.test)
```

```
## [1] 14718    54
```

```
## [1] 4904   54
```

The dependent variable is classe and it's distribution in the training and in the training,train set is shown in the plot.

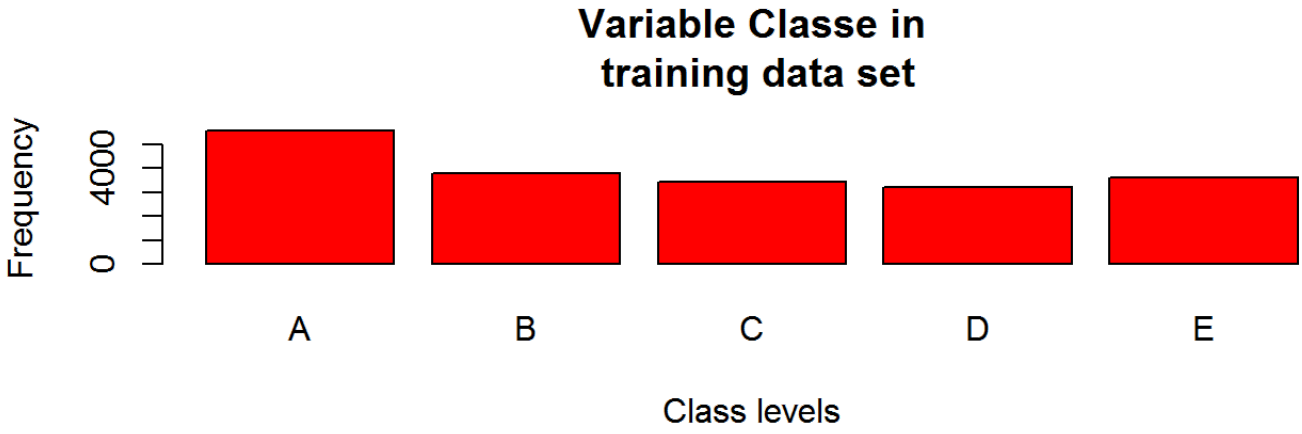
```
table(training.data$classe)
```

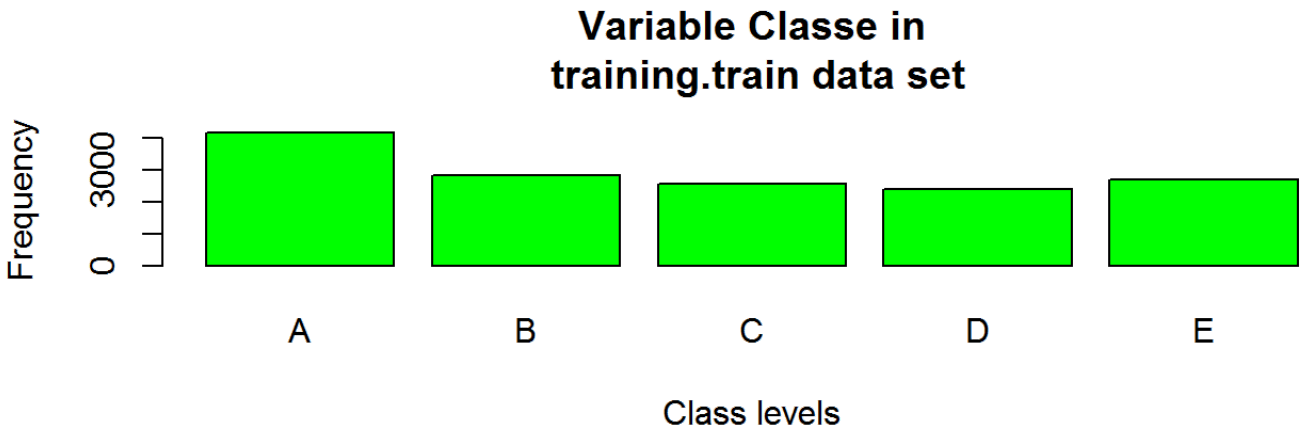
```
##
##    A    B    C    D    E
## 5580 3797 3422 3216 3607
```

```
table(training.train$classe)
```

```
##
##    A    B    C    D    E
## 4185 2848 2567 2412 2706
```

```
par(mfrow=c(2,1))
plot(training.data$classe,col="red",main="Variable Classe in\n training data set",
     ylab="Frequency",xlab="Class levels")
plot(training.train$classe,col="green",main="Variable Classe in\n training.train data set",
     ylab="Frequency",xlab="Class levels")
```





Since the outcome is categorical and the explanatory variables are all continuous random variables, the prediction model is built using different models using the subsetting training data. The model is then applied to the subsetted testing data to estimate the confusion matrix and consequently the accuracy. Showing this accuracy for all models is performed and the model with the best accuracy is selected.

Now we fit a model with: i) trees. ii) Random forests. iia) Random forests. iii) lda. iv) Naive-Bayes. v) SVM. vi) KSVM.

```
# 1. TREES
library(rpart)
tree <- rpart(classe~., data=training.train,method="class")
predict.tree <- predict(tree,type="class",newdata=training.test)
cmatrix.tree <- confusionMatrix(predict.tree,training.test$classe)



# 2. RANDOM FORESTS
library(randomForest)
```

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

```
rf <- randomForest(classe ~. , data=training.train, method="class")
prediction.rf <- predict(rf, training.test, type = "class")
cmatrix.rf <- confusionMatrix(prediction.rf, training.test$classe)



# 2a. RANDOM FORESTS, IMPORTANCE=TRUE
library(randomForest)
rf1 <- randomForest(classe ~. , data=training.train, importance=TRUE)
prediction.rf1 <- predict(rf1, training.test, type = "class")
cmatrix.rf1 <- confusionMatrix(prediction.rf1, training.test$classe)

# 3. LDA
lda <- train(classe~., data=training.train,method="lda")
```

```
## Loading required package: MASS
```

```
predict.lda <- predict(lda,newdata=training.test)
cmatrix.lda <- confusionMatrix(training.test$classe,predict.lda)

# 4. NAIVE-BAYES
library(klaR)
nb <- NaiveBayes(as.factor(classe)~., data=training.train)
predict.nb <- suppressWarnings(predict(nb,newdata=training.test))
cmatrix.nb <- confusionMatrix(training.test$classe,predict.nb$class)

# 5. SVM
library(e1071)
# tune does not end in a reasonable period of time, e.g. 4 hours
#tune.out <- tune(svm, classe ~. , data=training.train, kernel="radial",
#    ranges = list(cost=c(0.1, 1, 10, 100, 1000),
#    gamma=c(0.5, 1, 2, 3, 4)))
svmfit <- svm(classe ~. , data=training.train, kernel="radial", cost=10, gamma=1)
prediction.svm <- predict(svmfit, training.test, type = "class")
cmatrix.svm <- confusionMatrix(prediction.svm, training.test$classe)

# 6. KSVM
library(kernlab)
ksvmfit <- ksvm(classe ~. , data=training.train, kernel="rbfdot", kpar = "automatic", C = 60, cros
s = 2)
```

```
## Using automatic sigma estimation (sigest) for RBF or laplace kernel
```

```
prediction.ksvm <- predict(ksvmfit, training.test, type ="response")
cmatrix.ksvm <- confusionMatrix(prediction.ksvm, training.test$classe)

# Comparison of the models:
accuracy.tree <- round(as.numeric(cmatrix.tree$overall[1]),4)
accuracy.rf <- round(as.numeric(cmatrix.rf$overall[1]),4)
accuracy.lda <- round(as.numeric(cmatrix.lda$overall[1]),4)
accuracy.nb <- round(as.numeric(cmatrix.nb$overall[1]),4)
accuracy.rf1 <- round(as.numeric(cmatrix.rf1$overall[1]),4)
accuracy.svm <- round(as.numeric(cmatrix.svm$overall[1]),4)
accuracy.ksvm <- round(as.numeric(cmatrix.ksvm$overall[1]),4)
method <- c("Tree","Random Forest","LDA","Naive-Bayes", "Random Forest1", "SVM", "KSVM")
acc <- c(accuracy.tree,accuracy.rf,accuracy.lda,accuracy.nb, accuracy.rf1, accuracy.svm, accuracy.
ksvm)
a <- matrix(0,ncol=7,nrow=2)
a[1,] <- method
a[2,] <- acc
print(a)
```

```
##      [,1]     [,2]            [,3]     [,4]          [,5]
## [1,] "Tree"   "Random Forest" "LDA"    "Naive-Bayes" "Random Forest1"
## [2,] "0.7471" "0.9986"        "0.7104" "0.4992"      "0.9984"
##      [,6]     [,7]
## [1,] "SVM"    "KSVM"
## [2,] "0.9386" "0.9961"
```

```
# The model wich has the best fit is the Random Forest:
a[,2]
```

```
## [1] "Random Forest" "0.9986"
```

```
cmatrix.rf
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    0    0    0    0
##          B    0  948    4    0    0
##          C    0    1  851    1    0
##          D    0    0    0  803    1
##          E    0    0    0    0  900
##
## Overall Statistics
##
##                Accuracy : 0.9986
##                  95% CI : (0.9971, 0.9994)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9982
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9989   0.9953   0.9988   0.9989
## Specificity            1.0000   0.9990   0.9995   0.9998   1.0000
## Pos Pred Value         1.0000   0.9958   0.9977   0.9988   1.0000
## Neg Pred Value         1.0000   0.9997   0.9990   0.9998   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1933   0.1735   0.1637   0.1835
## Detection Prevalence   0.2845   0.1941   0.1739   0.1639   0.1835
## Balanced Accuracy      1.0000   0.9990   0.9974   0.9993   0.9994
```

The entire output of confusionMatrix for the best model (Random Forest) is shown above;it does provide the accuracy statistic as 0.9986, approximately 99.86%.

We now apply the best prediction model to the test dataset to predict the outcomes.

```
# Once we selected the best model, we  use it in test set.
rf.test <- predict(rf,testing.data,type="class")
rf.test
```

```
##  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
```

rf.test now contains the 20 predictions, which in order are: >B A B A A E D B A A B C B A E E A B B B

The remainder of the code simply writes these predictions to text files for uploading for assessment.

```
pml_write_files = function(x){
  n = length(x)
  for(i in 1:n){
    fname = paste0("problem_id_",i,".txt")
    write.table(x[i],file=fname,quote=FALSE,row.names=FALSE,col.names=FALSE)
  }
}
pml_write_files(rf.test)
```

References

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.

Read more: http://groupware.les.inf.puc-rio.br/har#ixzz3MTnYewf1 (http://groupware.les.inf.puc-rio.br/har#ixzz3MTnYewf1)