Machine Learning - Course Project

Aline Riquetti

1 de outubro de 2017

How you the model was built

The model was built with the intention of predict in which class an observation fits. The class is a factor variable named as “classe” and represent the answer of participants in a experiment where participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in 5 different fashions:

- exactly according to the specification (Class A)

- throwing the elbows to the front (Class B)

- lifting the dumbbell only halfway (Class C)

- lowering the dumbbell only halfway (Class D)

- throwing the hips to the front (Class E)

To predict which class an observation belongs to the other variables provided were used, as appropriate, applying some transformations in the data, if necessary.

Finally, 6 prediction algorithms were built, and from these, a last model was obtained based on the predictions obtained from the previously adjusted models;

How you the cross validation was used

Cross-validation was performed by sampling our training data set randomly without replacement into 2 subsamples:

-Training data (70% of the original Training data set)

-Testing data (30% of the original Training data set)

-Validation data (20 observation of of the original Testing data set)

Our models will be fitted on the Training data set, and tested on the Testing data. Once the combined predictors model was built, he was used to predict the original Testing dataset, now called validation dataset

Expected out-of-sample error

The expected out-of-sample error corresponded to the Accuracy in the cross-validation data.

Accuracy is the proportion of correct classified observation over the total sample in the Testing data set.

The choises made

To better fit the models some transformation are necessary in the original data.

Drop the variable with to many missing

Drop identifier variables

Drop variable with low variability

Transform variables highly correlated in two principal components

Finaly, the models were built. To obtain the best model, that is, with greater accuracy, 7 models was built. 6 of the using different algorithms, and the last model, using the prediction to finaly adjust the final result.

Results:

Load the necessaries packages

```
library(caret)
```

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

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

```r
library(gridExtra)
library(rpart)
library(rattle)
```

```
## Rattle: A free graphical interface for data science with R.
## Version 5.1.0 Copyright (c) 2006-2017 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```r
library(e1071)
library(klaR)
```

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

```r
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:rattle':
##
##     importance
```

```
## The following object is masked from 'package:gridExtra':
##
##     combine
```

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

Download the data

```r
urlTrain <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
urlTest <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

download.file(urlTrain, destfile = "pml-training.csv")
download.file(urlTest, destfile = "pml-testing.csv")
```

Read the data and make the study reproducible

The testing data will be user as validation data since the model build need a sample to test how accurately a predictive model is

```r
training_testing <- read.csv("pml-training.csv", na.strings=c("NA","#DIV/0!",""))
validation <- read.csv("pml-testing.csv", na.strings=c("NA","#DIV/0!",""))

set.seed(1235)
```

Partitioning the sample of data in training and testing

```r
inTrain <- createDataPartition(y=training_testing$classe, p=0.7, list=FALSE)
training <-training_testing[inTrain, ]
testing <- training_testing[-inTrain, ]
str(training)
```

```
## 'data.frame':    13737 obs. of  160 variables:
##  $ X                       : int  1 2 3 4 5 9 10 12 13 15 ...
##  $ 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 13230842
32 1323084232 1323084232 1323084232 ...
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 484323 484434 528316 560359 604281 ...
```

```
##  $ 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 1.43 1.45 1.43 1.42 1.45 ...
##  $ pitch_belt              : num  8.07 8.07 8.07 8.05 8.07 8.16 8.17 8.18 8.2 8.2 ...
##  $ 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 0.03 0.02 0.02 0 ...
##  $ 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 -0.02 0 0 ...
##  $ accel_belt_x            : int  -21 -22 -20 -22 -21 -20 -21 -22 -22 -21 ...
##  $ accel_belt_y            : int  4 4 5 3 2 2 4 2 4 2 ...
##  $ accel_belt_z            : int  22 22 23 21 24 24 22 23 21 22 ...
##  $ magnet_belt_x           : int  -3 -7 -2 -6 -6 1 -3 -2 -3 -1 ...
##  $ magnet_belt_y           : int  599 608 600 604 600 602 609 602 606 597 ...
##  $ magnet_belt_z           : int  -313 -311 -305 -310 -302 -312 -308 -319 -309 -310 ...
##  $ roll_arm                : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -129 ...
##  $ pitch_arm               : num  22.5 22.5 22.5 22.1 22.1 21.7 21.6 21.5 21.4 21.4 ...
##  $ 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 0.02 0.02 0.02 0.02 ...
##  $ gyros_arm_y             : num  0 -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -0.03 -0.02 0 ...
##  $ gyros_arm_z             : num  -0.02 -0.02 -0.02 0.02 0 -0.02 -0.02 0 -0.02 -0.03 ...
##  $ accel_arm_x             : int  -288 -290 -289 -289 -289 -288 -288 -288 -287 -289 ...
##  $ accel_arm_y             : int  109 110 110 111 111 109 110 111 111 111 ...
##  $ accel_arm_z             : int  -123 -125 -126 -123 -123 -122 -124 -123 -124 -124 ...
##  $ magnet_arm_x            : int  -368 -369 -368 -372 -374 -369 -376 -363 -372 -374 ...
##  $ magnet_arm_y            : int  337 337 344 344 337 341 334 343 338 342 ...
##  $ magnet_arm_z            : int  516 513 513 512 506 518 516 520 509 510 ...
##  $ 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]
```

```
dim(training); dim(testing)
```

```
## [1] 13737   160
```

```
## [1] 5885  160
```

Explore Target variable

```
counts <- table(training$classe)
barplot(counts, main="Training Data - Classe distribution", xlab="Number of observation",
col=c("darkblue","darkred","darkgreen", "orange", "purple"))
```

Cleaning the data

It's possible to see the data has a lot os missing values. Some variables can't be used because the proportion of missing values is too hight.

Lets keep only the variables that have missing proportion less than 30%

```
prop_missing<-sapply(training, function(x) sum(is.na(x))/nrow(training))
var_no_missing = data.frame(var=which(prop_missing<0.3, arr.ind=T))
training2 <- training[, var_no_missing$var]
dim(training2)
```

```
## [1] 13737    60
```

Eliminate no variability. Lets drop variable with no variability as well as the identification variables.

```
nearzero <- nearZeroVar(training2, saveMetrics = TRUE)
nearzero
```

```
##                      freqRatio percentUnique zeroVar   nzv
## X                     1.000000  100.00000000   FALSE FALSE
## user_name             1.127198    0.04367766   FALSE FALSE
## raw_timestamp_part_1  1.035714    6.09303341   FALSE FALSE
## raw_timestamp_part_2  1.000000   89.34265123   FALSE FALSE
## cvtd_timestamp        1.016023    0.14559220   FALSE FALSE
## new_window           48.592058    0.01455922   FALSE  TRUE
## num_window            1.000000    6.23862561   FALSE FALSE
## roll_belt             1.064725    8.07308728   FALSE FALSE
## pitch_belt            1.014599   12.26614253   FALSE FALSE
## yaw_belt              1.071225   12.99410352   FALSE FALSE
## total_accel_belt      1.074737    0.21110868   FALSE FALSE
## gyros_belt_x          1.075758    0.98274732   FALSE FALSE
## gyros_belt_y          1.143599    0.47317464   FALSE FALSE
## gyros_belt_z          1.061355    1.20113562   FALSE FALSE
## accel_belt_x          1.049270    1.15745796   FALSE FALSE
## accel_belt_y          1.104052    1.01186576   FALSE FALSE
## accel_belt_z          1.144558    2.08196841   FALSE FALSE
## magnet_belt_x         1.023810    2.26395865   FALSE FALSE
## magnet_belt_y         1.120000    2.10380724   FALSE FALSE
## magnet_belt_z         1.052308    3.14479144   FALSE FALSE
## roll_arm             53.340909   17.72584989   FALSE FALSE
## pitch_arm            90.269231   20.12812113   FALSE FALSE
## yaw_arm              33.056338   19.24000874   FALSE FALSE
## total_accel_arm       1.004702    0.46589503   FALSE FALSE
## gyros_arm_x           1.045977    4.59343379   FALSE FALSE
## gyros_arm_y           1.461318    2.67889641   FALSE FALSE
## gyros_arm_z           1.154519    1.70342870   FALSE FALSE
## accel_arm_x           1.000000    5.56890151   FALSE FALSE
## accel_arm_y           1.100000    3.82907476   FALSE FALSE
## accel_arm_z           1.166667    5.52522385   FALSE FALSE
## magnet_arm_x          1.112903    9.64548300   FALSE FALSE
## magnet_arm_y          1.030303    6.23134600   FALSE FALSE
## magnet_arm_z          1.051282    9.08495305   FALSE FALSE
## roll_dumbbell         1.147727   86.94038000   FALSE FALSE
## pitch_dumbbell        2.148515   84.52354954   FALSE FALSE
## yaw_dumbbell          1.147727   86.22697823   FALSE FALSE
## total_accel_dumbbell  1.107937    0.31302322   FALSE FALSE
## gyros_dumbbell_x      1.012077    1.72526753   FALSE FALSE
## gyros_dumbbell_y      1.203349    1.95821504   FALSE FALSE
## gyros_dumbbell_z      1.133005    1.41952391   FALSE FALSE
## accel_dumbbell_x      1.062500    3.00647885   FALSE FALSE
## accel_dumbbell_y      1.070588    3.29766325   FALSE FALSE
## accel_dumbbell_z      1.318182    2.96280119   FALSE FALSE
## magnet_dumbbell_x     1.032258    7.87653782   FALSE FALSE
## magnet_dumbbell_y     1.342105    5.97655966   FALSE FALSE
## magnet_dumbbell_z     1.196721    4.80454248   FALSE FALSE
## roll_forearm         11.535565   13.88221591   FALSE FALSE
## pitch_forearm        70.692308   18.86146903   FALSE FALSE
## yaw_forearm          15.930636   12.85579093   FALSE FALSE
## total_accel_forearm   1.106236    0.50229308   FALSE FALSE
## gyros_forearm_x       1.010695    2.03829075   FALSE FALSE
## gyros_forearm_y       1.101961    5.24131907   FALSE FALSE
## gyros_forearm_z       1.178886    2.10380724   FALSE FALSE
## accel_forearm_x       1.029851    5.64169761   FALSE FALSE
## accel_forearm_y       1.073529    7.13401762   FALSE FALSE
## accel_forearm_z       1.025424    4.07658150   FALSE FALSE
## magnet_forearm_x      1.000000   10.61367111   FALSE FALSE
## magnet_forearm_y      1.216667   13.34352479   FALSE FALSE
## magnet_forearm_z      1.023810   11.74201063   FALSE FALSE
## classe                1.469526    0.03639805   FALSE FALSE
```

```
table(training2$new_window)
```

```
##
##    no   yes
## 13460   277
```

```
training2 <- training2[,-6]
training2 <- training2[, -c(1:5)]
dim(training2)
```

```
## [1] 13737    54
```

Convert variable integer to numerics

```
nums <- sapply(training2, is.integer)
integer = data.frame(integ=which(nums==TRUE))
training2[integer$integ] <- lapply(training2[integer$integ], as.numeric)

str(training2); dim(training2)
```

```
## 'data.frame':    13737 obs. of  54 variables:
##  $ num_window          : num  11 11 11 12 12 12 12 12 12 12 ...
##  $ roll_belt           : num  1.41 1.41 1.42 1.48 1.48 1.43 1.45 1.43 1.42 1.45 ...
##  $ pitch_belt          : num  8.07 8.07 8.07 8.05 8.07 8.16 8.17 8.18 8.2 8.2 ...
##  $ 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    : num  3 3 3 3 3 3 3 3 3 3 ...
##  $ gyros_belt_x        : num  0 0.02 0 0.02 0.02 0.02 0.03 0.02 0.02 0 ...
##  $ 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 -0.02 0 0 ...
##  $ accel_belt_x        : num  -21 -22 -20 -22 -21 -20 -21 -22 -22 -21 ...
##  $ accel_belt_y        : num  4 4 5 3 2 2 4 2 4 2 ...
##  $ accel_belt_z        : num  22 22 23 21 24 24 22 23 21 22 ...
##  $ magnet_belt_x       : num  -3 -7 -2 -6 -6 1 -3 -2 -3 -1 ...
##  $ magnet_belt_y       : num  599 608 600 604 600 602 609 602 606 597 ...
##  $ magnet_belt_z       : num  -313 -311 -305 -310 -302 -312 -308 -319 -309 -310 ...
##  $ roll_arm            : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -129 ...
##  $ pitch_arm           : num  22.5 22.5 22.5 22.1 22.1 21.7 21.6 21.5 21.4 21.4 ...
##  $ yaw_arm             : num  -161 -161 -161 -161 -161 -161 -161 -161 -161 -161 ...
##  $ total_accel_arm     : num  34 34 34 34 34 34 34 34 34 34 ...
##  $ gyros_arm_x         : num  0 0.02 0.02 0.02 0 0.02 0.02 0.02 0.02 0.02 ...
##  $ gyros_arm_y         : num  0 -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -0.03 -0.02 0 ...
##  $ gyros_arm_z         : num  -0.02 -0.02 -0.02 0.02 0 -0.02 -0.02 0 -0.02 -0.03 ...
##  $ accel_arm_x         : num  -288 -290 -289 -289 -289 -288 -288 -288 -287 -289 ...
##  $ accel_arm_y         : num  109 110 110 111 111 109 110 111 111 111 ...
##  $ accel_arm_z         : num  -123 -125 -126 -123 -123 -122 -124 -123 -124 -124 ...
##  $ magnet_arm_x        : num  -368 -369 -368 -372 -374 -369 -376 -363 -372 -374 ...
##  $ magnet_arm_y        : num  337 337 344 344 337 341 334 343 338 342 ...
##  $ magnet_arm_z        : num  516 513 513 512 506 518 516 520 509 510 ...
##  $ 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: num  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 -0.02 0 ...
##  $ accel_dumbbell_x    : num  -234 -233 -232 -232 -233 -232 -235 -233 -234 -234 ...
##  $ accel_dumbbell_y    : num  47 47 46 48 48 47 48 47 48 47 ...
##  $ accel_dumbbell_z    : num  -271 -269 -270 -269 -270 -269 -270 -270 -269 -270 ...
##  $ magnet_dumbbell_x   : num  -559 -555 -561 -552 -554 -549 -558 -554 -552 -554 ...
##  $ magnet_dumbbell_y   : num  293 296 298 303 292 292 291 291 302 294 ...
##  $ magnet_dumbbell_z   : num  -65 -64 -63 -60 -68 -65 -69 -65 -69 -63 ...
##  $ roll_forearm        : num  28.4 28.3 28.3 28.1 28 27.7 27.7 27.5 27.2 27.2 ...
##  $ pitch_forearm       : num  -63.9 -63.9 -63.9 -63.9 -63.9 -63.8 -63.8 -63.8 -63.9 -63.9 ...
##  $ yaw_forearm         : num  -153 -153 -152 -152 -152 -152 -152 -152 -151 -151 ...
##  $ total_accel_forearm : num  36 36 36 36 36 36 36 36 36 36 ...
##  $ gyros_forearm_x     : num  0.03 0.02 0.03 0.02 0.02 0.03 0.02 0.02 0 0 ...
##  $ gyros_forearm_y     : num  0 0 -0.02 -0.02 0 0 0 0.02 0 -0.02 ...
##  $ gyros_forearm_z     : num  -0.02 -0.02 0 0 -0.02 -0.02 -0.02 -0.03 -0.03 -0.02 ...
##  $ accel_forearm_x     : num  192 192 196 189 189 193 190 191 193 192 ...
##  $ accel_forearm_y     : num  203 203 204 206 206 204 205 203 205 201 ...
##  $ accel_forearm_z     : num  -215 -216 -213 -214 -214 -214 -215 -215 -215 -214 ...
##  $ magnet_forearm_x    : num  -17 -18 -18 -16 -17 -16 -22 -11 -15 -16 ...
##  $ magnet_forearm_y    : num  654 661 658 658 655 653 656 657 655 656 ...
##  $ magnet_forearm_z    : num  476 473 469 469 473 476 473 478 472 472 ...
##  $ classe              : Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
## [1] 13737    54
```

Pricipal component analysis

Check variable that are highly correlated with each other. It means a correlation coefeciecient greater than 0.9

```
Cor<-abs(cor(training2[,c(-54)]))
diag(Cor)<-0
correlation = data.frame(which(Cor>0.90, arr.ind=T))
```

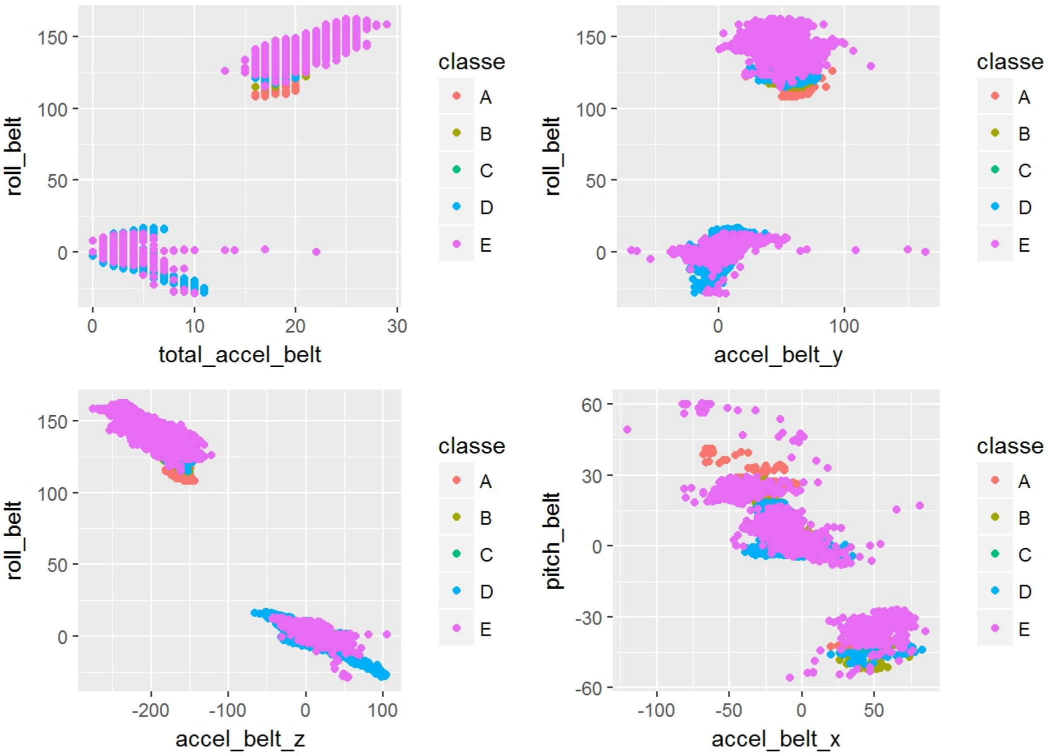
```
## Warning in data.row.names(row.names, rowsi, i): some row.names duplicated:
## 6,7,9,10,11,12,13,14,20,21,22 --> row.names NOT used
```

```
cor<-unique(correlation[,"row"])
which(Cor>0.90, arr.ind=T)
```

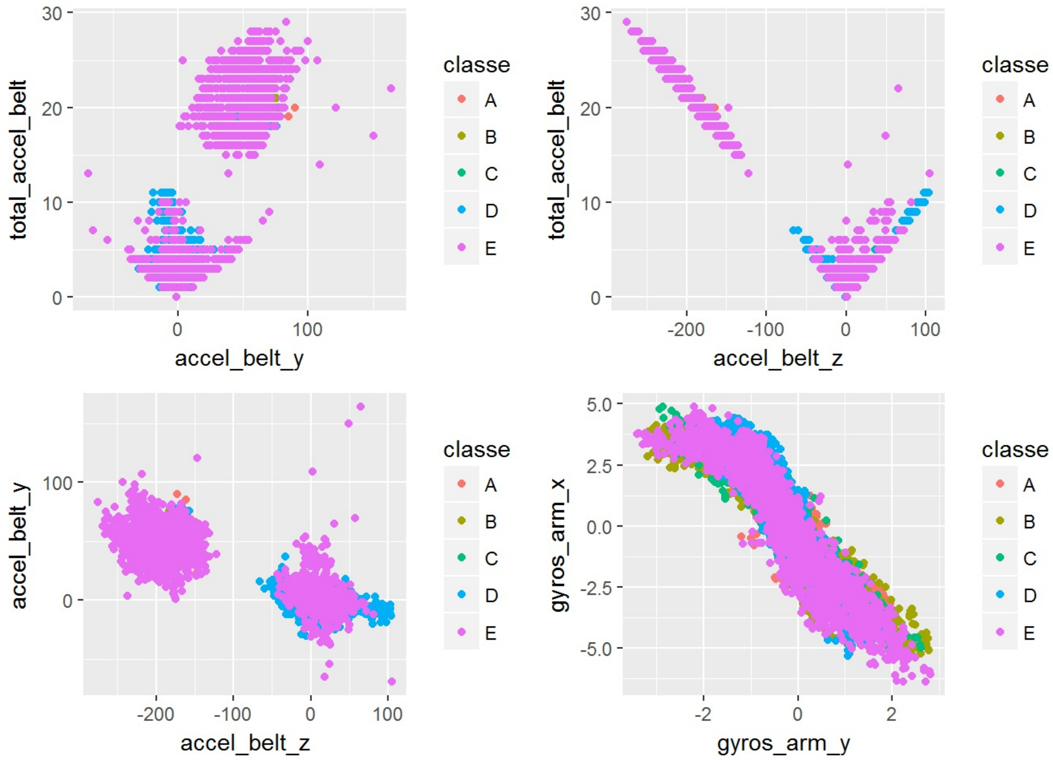
```
##                  row col
## total_accel_belt   5   2
## accel_belt_y      10   2
## accel_belt_z      11   2
## accel_belt_x       9   3
## roll_belt          2   5
## accel_belt_y      10   5
## accel_belt_z      11   5
## pitch_belt         3   9
## roll_belt          2  10
## total_accel_belt   5  10
## accel_belt_z      11  10
## roll_belt          2  11
## total_accel_belt   5  11
## accel_belt_y      10  11
## gyros_arm_y       20  19
## gyros_arm_x       19  20
## gyros_dumbbell_z  34  32
## gyros_forearm_z   47  32
## gyros_dumbbell_x  32  34
## gyros_forearm_z   47  34
## gyros_dumbbell_x  32  47
## gyros_dumbbell_z  34  47
```

Plot some ghaphs to se how correlated some of theses variables are

```
plot1<-qplot(total_accel_belt, roll_belt, colour=classe, data=training2)
plot2<-qplot(accel_belt_y, roll_belt, colour=classe, data=training2)
plot3<-qplot(accel_belt_z, roll_belt, colour=classe, data=training2)
plot4<-qplot(accel_belt_x, pitch_belt, colour=classe, data=training2)
grid.arrange(plot1, plot2, plot3, plot4, ncol=2)
```

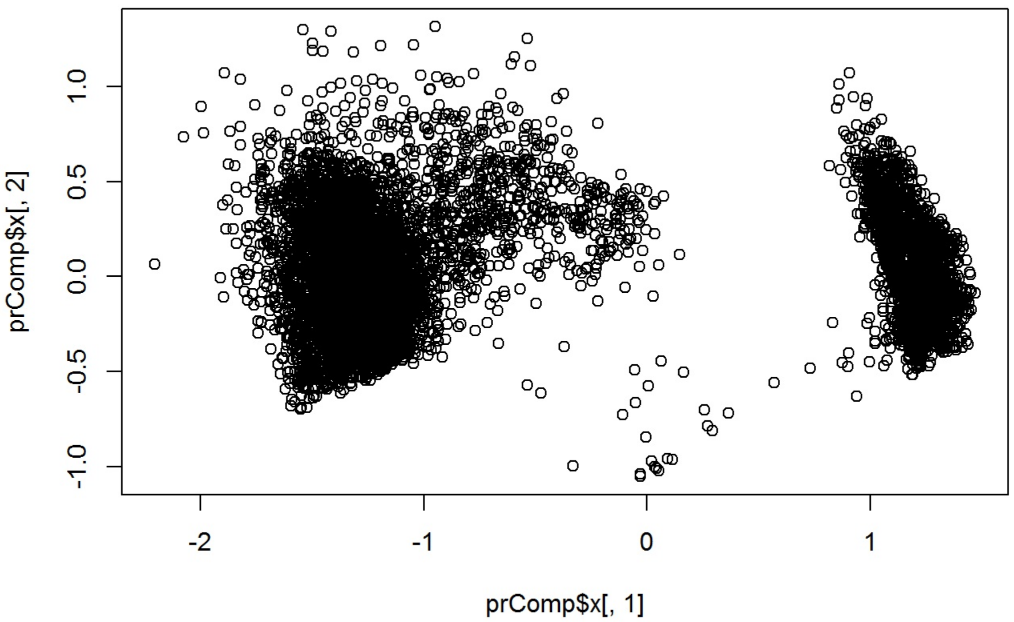



```
plot5<-qplot(accel_belt_y, total_accel_belt, colour=classe, data=training2)
plot6<-qplot(accel_belt_z, total_accel_belt, colour=classe, data=training2)
plot7<-qplot(accel_belt_z, accel_belt_y, colour=classe, data=training2)
plot8<-qplot(gyros_arm_y, gyros_arm_x, colour=classe, data=training2)
grid.arrange(plot5, plot6, plot7, plot8, ncol=2)
```



To avoid multicolinearity lets use PCA only for the variables that had high correlation with the others, reducing the data dimension

```
prComp<-prcomp(log10(abs(training2[,cor])+1))
plot(prComp$x[,1], prComp$x[,2])
```




```
preProc<-preProcess(log10(abs(training2[,cor])+1),method="pca",pcaComp=2)
PCA_train<-predict(preProc, log10(abs(training2[,cor])+1))
training3<-cbind(training2[,-cor],PCA_train)
str(training3); dim(training3)
```

```
## 'data.frame':    13737 obs. of  45 variables:
##  $ num_window          : num  11 11 11 12 12 12 12 12 12 12 ...
##  $ yaw_belt            : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 ...
##  $ gyros_belt_x        : num  0 0.02 0 0.02 0.02 0.02 0.03 0.02 0.02 0 ...
##  $ 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 -0.02 0 0 ...
##  $ magnet_belt_x       : num  -3 -7 -2 -6 -6 1 -3 -2 -3 -1 ...
##  $ magnet_belt_y       : num  599 608 600 604 600 602 609 602 606 597 ...
##  $ magnet_belt_z       : num  -313 -311 -305 -310 -302 -312 -308 -319 -309 -310 ...
##  $ roll_arm            : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -129 ...
##  $ pitch_arm           : num  22.5 22.5 22.5 22.1 22.1 21.7 21.6 21.5 21.4 21.4 ...
##  $ yaw_arm             : num  -161 -161 -161 -161 -161 -161 -161 -161 -161 -161 ...
##  $ total_accel_arm     : num  34 34 34 34 34 34 34 34 34 34 ...
##  $ gyros_arm_z         : num  -0.02 -0.02 -0.02 0.02 0 -0.02 -0.02 0 -0.02 -0.03 ...
##  $ accel_arm_x         : num  -288 -290 -289 -289 -289 -288 -288 -288 -287 -289 ...
##  $ accel_arm_y         : num  109 110 110 111 111 109 110 111 111 111 ...
##  $ accel_arm_z         : num  -123 -125 -126 -123 -123 -122 -124 -123 -124 -124 ...
##  $ magnet_arm_x        : num  -368 -369 -368 -372 -374 -369 -376 -363 -372 -374 ...
##  $ magnet_arm_y        : num  337 337 344 344 337 341 334 343 338 342 ...
##  $ magnet_arm_z        : num  516 513 513 512 506 518 516 520 509 510 ...
##  $ 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: num  37 37 37 37 37 37 37 37 37 37 ...
##  $ gyros_dumbbell_y    : num  -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 ...
##  $ accel_dumbbell_x    : num  -234 -233 -232 -232 -233 -232 -235 -233 -234 -234 ...
##  $ accel_dumbbell_y    : num  47 47 46 48 48 47 48 47 48 47 ...
##  $ accel_dumbbell_z    : num  -271 -269 -270 -269 -270 -269 -270 -270 -269 -270 ...
##  $ magnet_dumbbell_x   : num  -559 -555 -561 -552 -554 -549 -558 -554 -552 -554 ...
##  $ magnet_dumbbell_y   : num  293 296 298 303 292 292 291 291 302 294 ...
##  $ magnet_dumbbell_z   : num  -65 -64 -63 -60 -68 -65 -69 -65 -69 -63 ...
##  $ roll_forearm        : num  28.4 28.3 28.3 28.1 28 27.7 27.7 27.5 27.2 27.2 ...
##  $ pitch_forearm       : num  -63.9 -63.9 -63.9 -63.9 -63.9 -63.8 -63.8 -63.8 -63.9 -63.9 ...
##  $ yaw_forearm         : num  -153 -153 -152 -152 -152 -152 -152 -152 -151 -151 ...
##  $ total_accel_forearm : num  36 36 36 36 36 36 36 36 36 36 ...
##  $ gyros_forearm_x     : num  0.03 0.02 0.03 0.02 0.02 0.03 0.02 0.02 0 0 ...
##  $ gyros_forearm_y     : num  0 0 -0.02 -0.02 0 0 0 0.02 0 -0.02 ...
##  $ accel_forearm_x     : num  192 192 196 189 189 193 190 191 193 192 ...
##  $ accel_forearm_y     : num  203 203 204 206 206 204 205 203 205 201 ...
##  $ accel_forearm_z     : num  -215 -216 -213 -214 -214 -214 -215 -215 -215 -214 ...
##  $ magnet_forearm_x    : num  -17 -18 -18 -16 -17 -16 -22 -11 -15 -16 ...
##  $ magnet_forearm_y    : num  654 661 658 658 655 653 656 657 655 656 ...
##  $ magnet_forearm_z    : num  476 473 469 469 473 476 473 478 472 472 ...
##  $ classe              : Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ PC1                 : num  -1.65 -1.62 -1.6 -1.71 -1.75 ...
##  $ PC2                 : num  3.1 3.05 3.06 3.02 3.06 ...
```

```
## [1] 13737    45
```

Without look to testing dataset lets make the same transformation we did for training dataset

```
testing2 <- testing[, var_no_missing$var]
testing2 <- testing2[,-6]
testing2 <- testing2[, -c(1:5)]
testing2[integer$integ] <- lapply(testing2[integer$integ], as.numeric)
PCA_test<-predict(preProc, log10(abs(testing2[,cor])+1))
testing3<-cbind(testing2[,-cor],PCA_test)
```

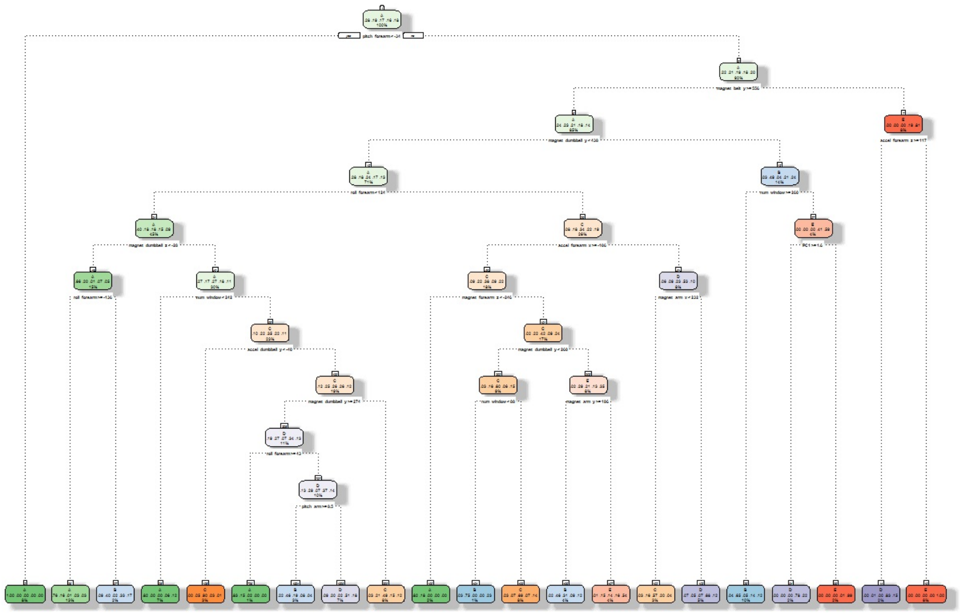
Due to the poor performance obtained to adjust the models, the default parameters of the cross validation have been changed. The method of cross-validation was maintained, with the number of interactions being altered and allowing parallelism

```
options <- trainControl(method = "cv", number = 7, allowParallel=TRUE)
```

Adjusting different prediction algorithms.

Decision tree

```
modFitA <- rpart(training3$classe ~ ., data=training3, method="class")
fancyRpartPlot(modFitA)
```



Rattle 2017-out-01 17:02:35 Administrador

```
predFitA<-predict(modFitA, testing3, type = "class")
mA<-confusionMatrix(testing3$classe, predFitA)
```

Random Forest

```
modFitB <- randomForest(classe ~. , data= training3)
predFitB<-predict(modFitB, testing3)
mB<-confusionMatrix(testing3$classe, predFitB)
```

Boosting

```
modFitC <- train(classe ~ ., data = training3, method = "LogitBoost", trControl= options)
predFitC<-predict(modFitC, testing3)
mC<-confusionMatrix(testing3$classe, predFitC)
```

Linear Descriminant Analysis

```
modFitD <- train(classe~., data=training3, method="lda", trControl= options)
predFitD<-predict(modFitD, testing3)
mD<-confusionMatrix(testing3$classe, predFitD)
```

Naive Bayes

```
modFitE <- train(classe~., data=training3, method="nb", trControl= options)
predFitE<-predict(modFitE, testing3)
mE<-confusionMatrix(testing3$classe, predFitE)
```

Bagging

```
modFitF <- train(classe~., data=training3, method="treebag", trControl= options)
predFitF<-predict(modFitF, testing3)
mF<-confusionMatrix(testing3$classe, predFitF)
```

Comparing results of algorithms

```
algorithm <- c("Decision Tree", "Random Forest","LogitBoost","Linear Descriminant Analysis","Naive Bayes", "Bag
ging")
Accuracy <- c(mA$overall['Accuracy'],  mB$overall['Accuracy'],   mC$overall['Accuracy'],
              mD$overall['Accuracy'],  mE$overall['Accuracy'],   mF$overall['Accuracy'])
results <- cbind(algorithm,Accuracy)
results
```

```
##          algorithm                          Accuracy
## Accuracy "Decision Tree"                    "0.688870008496177"
## Accuracy "Random Forest"                    "0.9964316057774"
## Accuracy "LogitBoost"                       "0.928878468151622"
## Accuracy "Linear Descriminant Analysis" "0.678164825828377"
## Accuracy "Naive Bayes"                      "0.763296516567545"
## Accuracy "Bagging"                          "0.990314358538658"
```

Combining predictors

Build a dataset with the results of predictions.

Boosting won't be used because generated a lot os missing value

```
combDF<-data.frame(predA=predFitA, predB=predFitB, predD=predFitD,
                   predE=predFitE, predF=predFitF, classe=testing3$classe)
```

Use the Random Forest to adjust a model based in the result of the previous models (modFitComb)

```
modFitComb <- randomForest(classe ~. , data= combDF)
```

Without look to validation dataset lets make the same transformation we did for training dataset

```
validation2 <- validation[, var_no_missing$var]
validation2 <- validation2[,-6]
validation2 <- validation2[, -c(1:5)]
validation2[integer$integ] <- lapply(validation2[integer$integ], as.numeric)
PCA_valid<-predict(preProc, log10(abs(validation2[,cor])+1))
validation3<-cbind(validation2[,-cor],PCA_valid)
```

Builing a dataset for validation with the prediction of each algorithm

```
predAV<-predict(modFitA, validation3, type = "class");
predBV<-predict(modFitB, validation3);
predCV<-predict(modFitC, validation3); predDV<-predict(modFitD, validation3);
predEV<-predict(modFitE, validation3); predFV<-predict(modFitF, validation3);
CombValid<-data.frame(predA=predAV, predB=predBV, predD=predDV, predE=predEV, predF=predFV)
```

Using the combined predictors (modFitComb) to predict classe for validation dataset

```
predFitComb<-predict(modFitComb, CombValid)
predFitComb
```

```
##  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
```