Course Project - Practical Machine Learning

Vergara, Enrico Marco

June 17, 2019

I. Overview

A. Background

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 Ã¢ÂÂ 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.

B. Data Definition

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)

Title: Weight Lifting Exercises Dataset

Basic Summary: This human activity recognition research has traditionally focused on discriminating between different activities, i.e. to predict “which” activity was performed at a specific point in time. The approach we propose for the Weight Lifting Exercises dataset is to investigate “how (well)” an activity was performed by the wearer. The “how (well)” investigation has only received little attention so far, even though it potentially provides useful information for a large variety of applications,such as sports training.

Six young health participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five 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) and throwing the hips to the front (Class E).

Source: *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.*

II. Data Pre-processing and Correlation Analysis

A. Data Loading

In this section, the initial processing of data is provided. The first thing to do is to download and load the data frame by storing it into a variable.

```
url_train <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
url_test  <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
pml_train <- read.csv(url(url_train))
pml_validation  <- read.csv(url(url_test))
```

B. Data Partitioning

For the prediction model, the training data is to be splitted into the “ideal” ratio of data partition for training and testing which is 70% as train data and 30% test data. This splitted data will also be used for the computation of the out-of-sample errors.

```
set.seed(123456789)
partition  <- createDataPartition(pml_train$classe, p=0.7, list=FALSE)
train_set <- pml_train[partition, ]
test_set  <- pml_train[-partition, ]

dim(train_set)
```

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

```
dim(test_set)
```

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

The training data set is made of **13737 observations** on **160 variables**. On the other hand, the testing data set is composed of **5885 observations** on **160 variables**.

```
str(train_set)
```

```
## 'data.frame':    13737 obs. of  160 variables:
##  $ X                       : int  3 4 6 7 8 9 10 11 12 13 ...
##  $ user_name               : Factor w/ 6 levels "adelmo","carlitos",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ raw_timestamp_part_1    : int  1323084231 1323084232 1323084232 1323084232 1323084232 1323084232 1323
084232 1323084232 1323084232 1323084232 ...
##  $ raw_timestamp_part_2    : int  820366 120339 304277 368296 440390 484323 484434 500302 528316 56035
9 ...
##  $ 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 12 12 12 12 12 12 12 12 12 ...
##  $ roll_belt               : num  1.42 1.48 1.45 1.42 1.42 1.43 1.45 1.45 1.43 1.42 ...
##  $ pitch_belt              : num  8.07 8.05 8.06 8.09 8.13 8.16 8.17 8.18 8.18 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      : Factor w/ 397 levels "","-0.016850",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_picth_belt     : Factor w/ 317 levels "","-0.021887",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_yaw_belt       : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_belt      : Factor w/ 395 levels "","-0.003095",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_belt.1    : Factor w/ 338 levels "","-0.005928",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_yaw_belt       : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ 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            : Factor w/ 68 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ 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            : Factor w/ 68 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ 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      : Factor w/ 4 levels "","#DIV/0!","0.00",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ 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 0.02 0.02 0.02 0.02 0.03 0.03 0.02 0.02 ...
##  $ gyros_belt_y            : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ gyros_belt_z            : num  -0.02 -0.03 -0.02 -0.02 -0.02 -0.02 0 -0.02 -0.02 0 ...
##  $ accel_belt_x            : int  -20 -22 -21 -22 -22 -20 -21 -21 -22 -22 ...
##  $ accel_belt_y            : int  5 3 4 3 4 2 4 2 2 4 ...
##  $ accel_belt_z            : int  23 21 21 21 21 24 22 23 23 21 ...
##  $ magnet_belt_x           : int  -2 -6 0 -4 -2 1 -3 -5 -2 -3 ...
##  $ magnet_belt_y           : int  600 604 603 599 603 602 609 596 602 606 ...
##  $ magnet_belt_z           : int  -305 -310 -312 -311 -313 -312 -308 -317 -319 -309 ...
##  $ roll_arm                : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -128 ...
##  $ pitch_arm               : num  22.5 22.1 22 21.9 21.8 21.7 21.6 21.5 21.5 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  0.02 0.02 0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 ...
##  $ gyros_arm_y             : num  -0.02 -0.03 -0.03 -0.03 -0.02 -0.03 -0.03 -0.03 -0.03 -0.02 ...
##  $ gyros_arm_z             : num  -0.02 0.02 0 0 0 -0.02 -0.02 0 0 -0.02 ...
##  $ accel_arm_x             : int  -289 -289 -289 -289 -289 -288 -288 -290 -288 -287 ...
##  $ accel_arm_y             : int  110 111 111 111 111 109 110 110 111 111 ...
##  $ accel_arm_z             : int  -126 -123 -122 -125 -124 -122 -124 -123 -123 -124 ...
##  $ magnet_arm_x            : int  -368 -372 -369 -373 -372 -369 -376 -366 -363 -372 ...
##  $ magnet_arm_y            : int  344 344 342 336 338 341 334 339 343 338 ...
##  $ magnet_arm_z            : int  513 512 513 509 510 518 516 509 520 509 ...
##  $ kurtosis_roll_arm       : Factor w/ 330 levels "","-0.02438",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_picth_arm      : Factor w/ 328 levels "","-0.00484",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_yaw_arm        : Factor w/ 395 levels "","-0.01548",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_arm       : Factor w/ 331 levels "","-0.00051",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_pitch_arm      : Factor w/ 328 levels "","-0.00184",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_yaw_arm        : Factor w/ 395 levels "","-0.00311",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ 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  12.9 13.4 13.4 13.1 12.8 ...
##  $ pitch_dumbbell          : num  -70.3 -70.4 -70.8 -70.2 -70.3 ...
##  $ yaw_dumbbell            : num  -85.1 -84.9 -84.5 -85.1 -85.1 ...
##  $ kurtosis_roll_dumbbell  : Factor w/ 398 levels "","-0.0035","-0.0073",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_picth_dumbbell : Factor w/ 401 levels "","-0.0163","-0.0233",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_yaw_dumbbell   : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_roll_dumbbell  : Factor w/ 401 levels "","-0.0082","-0.0096",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_pitch_dumbbell : Factor w/ 402 levels "","-0.0053","-0.0084",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_yaw_dumbbell   : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1 1 1 1 ...
##  $ 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        : Factor w/ 73 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ 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        : Factor w/ 73 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ amplitude_roll_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##   [list output truncated]
```

From the summary, it is noticeable that many columns have NA values or blank values on almost every observation. This is an indication of irrelevant featuresm thus it is safe to consider removing them. The behavior is pretty much similar for both testing and training set. Thus what will be applied to training in terms of cleaning will be also applied to testing.

C. Data Cleaning

a. Definitive Variables

The first seven columns give information about the people who did the test, and also timestamps. This are again irrelevant for the model. So the first thing to consider is removing this variables.

```
train_set_clean <- train_set[,-c(1:7)]
test_set_clean <- test_set[,-c(1:7)]
```

b. Near Zero Covariates

It is highly emphasized that if there are near zero variables in the data. It is just proper to removed them since it only makes the model bias and inaccurate.

```
nzv <- nearZeroVar(train_set_clean,saveMetrics=TRUE)
train_set_clean <- train_set_clean[, nzv$nzv==FALSE]
test_set_clean <- test_set_clean[, nzv$nzv==FALSE]
nzv
```

```
##                          freqRatio percentUnique zeroVar   nzv
## roll_belt                 1.034375    8.08764650   FALSE FALSE
## pitch_belt                1.204918   12.17150761   FALSE FALSE
## yaw_belt                  1.077348   13.15425493   FALSE FALSE
## total_accel_belt          1.077358    0.21110868   FALSE FALSE
## kurtosis_roll_belt     1681.000000    2.05284997   FALSE  TRUE
## kurtosis_picth_belt     707.789474    1.71798792   FALSE  TRUE
## kurtosis_yaw_belt        46.532872    0.01455922   FALSE  TRUE
## skewness_roll_belt     1921.142857    2.05284997   FALSE  TRUE
## skewness_roll_belt.1    707.789474    1.81262284   FALSE  TRUE
## skewness_yaw_belt        46.532872    0.01455922   FALSE  TRUE
## max_roll_belt             1.142857    1.15745796   FALSE FALSE
## max_picth_belt            1.625000    0.15287181   FALSE FALSE
## max_yaw_belt            517.230769    0.40765815   FALSE  TRUE
## min_roll_belt             1.125000    1.12105991   FALSE FALSE
## min_pitch_belt            2.157895    0.10919415   FALSE FALSE
## min_yaw_belt            517.230769    0.40765815   FALSE  TRUE
## amplitude_roll_belt       1.304348    0.85899396   FALSE FALSE
## amplitude_pitch_belt      3.276596    0.09463493   FALSE FALSE
## amplitude_yaw_belt       49.260073    0.02911844   FALSE  TRUE
## var_total_accel_belt      1.482143    0.40765815   FALSE FALSE
## avg_roll_belt             1.090909    1.13561913   FALSE FALSE
## stddev_roll_belt          1.000000    0.45861542   FALSE FALSE
## var_roll_belt             1.648148    0.56780957   FALSE FALSE
## avg_pitch_belt            1.000000    1.26665211   FALSE FALSE
## stddev_pitch_belt         1.086957    0.28390478   FALSE FALSE
## var_pitch_belt            1.159420    0.40765815   FALSE FALSE
## avg_yaw_belt              1.125000    1.37584625   FALSE FALSE
## stddev_yaw_belt           1.875000    0.37126010   FALSE FALSE
## var_yaw_belt              1.354839    0.87355318   FALSE FALSE
## gyros_belt_x              1.072034    0.92451045   FALSE FALSE
## gyros_belt_y              1.153819    0.46589503   FALSE FALSE
## gyros_belt_z              1.064205    1.20113562   FALSE FALSE
## accel_belt_x              1.068519    1.17201718   FALSE FALSE
## accel_belt_y              1.122109    0.99730654   FALSE FALSE
## accel_belt_z              1.055921    2.11836646   FALSE FALSE
## magnet_belt_x             1.108871    2.21300138   FALSE FALSE
## magnet_belt_y             1.103604    2.08196841   FALSE FALSE
## magnet_belt_z             1.002924    3.15207105   FALSE FALSE
## roll_arm                 47.560000   17.36186940   FALSE FALSE
## pitch_arm                76.741935   20.28827255   FALSE FALSE
## yaw_arm                  30.487179   19.10897576   FALSE FALSE
## total_accel_arm           1.079872    0.48045425   FALSE FALSE
## var_accel_arm             7.000000    2.06012958   FALSE FALSE
## avg_roll_arm             49.000000    1.75438596   FALSE  TRUE
## stddev_roll_arm          49.000000    1.75438596   FALSE  TRUE
## var_roll_arm             49.000000    1.75438596   FALSE  TRUE
## avg_pitch_arm            49.000000    1.75438596   FALSE  TRUE
## stddev_pitch_arm         49.000000    1.75438596   FALSE  TRUE
## var_pitch_arm            49.000000    1.75438596   FALSE  TRUE
## avg_yaw_arm              49.000000    1.75438596   FALSE  TRUE
## stddev_yaw_arm           51.000000    1.73982675   FALSE  TRUE
## var_yaw_arm              51.000000    1.73982675   FALSE  TRUE
## gyros_arm_x               1.010753    4.61527262   FALSE FALSE
## gyros_arm_y               1.490411    2.66433719   FALSE FALSE
## gyros_arm_z               1.056848    1.70342870   FALSE FALSE
## accel_arm_x               1.133929    5.57618112   FALSE FALSE
## accel_arm_y               1.165605    3.79995632   FALSE FALSE
## accel_arm_z               1.068182    5.60529956   FALSE FALSE
```

```
## magnet_arm_x               1.101695    9.65276261   FALSE FALSE
## magnet_arm_y               1.016129    6.22406639   FALSE FALSE
## magnet_arm_z               1.012987    9.11407149   FALSE FALSE
## kurtosis_roll_arm        268.960000    1.75438596   FALSE  TRUE
## kurtosis_picth_arm       263.686275    1.74710636   FALSE  TRUE
## kurtosis_yaw_arm        1921.142857    2.06012958   FALSE  TRUE
## skewness_roll_arm        274.448980    1.76166557   FALSE  TRUE
## skewness_pitch_arm       263.686275    1.74710636   FALSE  TRUE
## skewness_yaw_arm        1921.142857    2.06740919   FALSE  TRUE
## max_roll_arm              16.333333    1.60151416   FALSE FALSE
## max_picth_arm              8.166667    1.50687923   FALSE FALSE
## max_yaw_arm                1.058824    0.36398049   FALSE FALSE
## min_roll_arm              16.333333    1.55783650   FALSE FALSE
## min_pitch_arm             12.250000    1.55783650   FALSE FALSE
## min_yaw_arm                1.052632    0.26206595   FALSE FALSE
## amplitude_roll_arm        24.500000    1.66703065   FALSE  TRUE
## amplitude_pitch_arm       17.000000    1.63791221   FALSE FALSE
## amplitude_yaw_arm          1.187500    0.36398049   FALSE FALSE
## roll_dumbbell              1.009804   86.31433355   FALSE FALSE
## pitch_dumbbell             2.000000   84.32700007   FALSE FALSE
## yaw_dumbbell               1.133333   85.69556672   FALSE FALSE
## kurtosis_roll_dumbbell  3362.000000    2.06740919   FALSE  TRUE
## kurtosis_picth_dumbbell 6724.000000    2.08196841   FALSE  TRUE
## kurtosis_yaw_dumbbell     46.532872    0.01455922   FALSE  TRUE
## skewness_roll_dumbbell  4482.666667    2.08924802   FALSE  TRUE
## skewness_pitch_dumbbell 6724.000000    2.08196841   FALSE  TRUE
## skewness_yaw_dumbbell     46.532872    0.01455922   FALSE  TRUE
## max_roll_dumbbell          1.333333    1.78350440   FALSE FALSE
## max_picth_dumbbell         1.000000    1.82718206   FALSE FALSE
## max_yaw_dumbbell         896.533333    0.46589503   FALSE  TRUE
## min_roll_dumbbell          1.000000    1.79806362   FALSE FALSE
## min_pitch_dumbbell         1.000000    1.91453738   FALSE FALSE
## min_yaw_dumbbell         896.533333    0.46589503   FALSE  TRUE
## amplitude_roll_dumbbell    7.000000    1.99461309   FALSE FALSE
## amplitude_pitch_dumbbell   7.000000    1.96549465   FALSE FALSE
## amplitude_yaw_dumbbell    47.185965    0.02183883   FALSE  TRUE
## total_accel_dumbbell       1.074816    0.30574361   FALSE FALSE
## var_accel_dumbbell         5.000000    1.97277426   FALSE FALSE
## avg_roll_dumbbell          1.000000    2.05284997   FALSE FALSE
## stddev_roll_dumbbell      14.000000    2.00917231   FALSE FALSE
## var_roll_dumbbell         14.000000    2.00917231   FALSE FALSE
## avg_pitch_dumbbell         1.000000    2.05284997   FALSE FALSE
## stddev_pitch_dumbbell     14.000000    2.00917231   FALSE FALSE
## var_pitch_dumbbell        14.000000    2.00917231   FALSE FALSE
## avg_yaw_dumbbell           1.000000    2.05284997   FALSE FALSE
## stddev_yaw_dumbbell       14.000000    2.00917231   FALSE FALSE
## var_yaw_dumbbell          14.000000    2.00917231   FALSE FALSE
## gyros_dumbbell_x           1.006961    1.71070831   FALSE FALSE
## gyros_dumbbell_y           1.286064    1.92181699   FALSE FALSE
## gyros_dumbbell_z           1.058962    1.43408313   FALSE FALSE
## accel_dumbbell_x           1.031111    2.95552158   FALSE FALSE
## accel_dumbbell_y           1.052023    3.27582442   FALSE FALSE
## accel_dumbbell_z           1.159509    2.89728471   FALSE FALSE
## magnet_dumbbell_x          1.068376    7.78918250   FALSE FALSE
## magnet_dumbbell_y          1.257812    6.02023732   FALSE FALSE
## magnet_dumbbell_z          1.021429    4.79726287   FALSE FALSE
## roll_forearm              11.118367   13.62742957   FALSE FALSE
## pitch_forearm             61.886364   18.99978161   FALSE FALSE
## yaw_forearm               16.106509   12.90674820   FALSE FALSE
```

```
## kurtosis_roll_forearm     203.757576    1.63791221   FALSE  TRUE
## kurtosis_picth_forearm    200.716418    1.63063260   FALSE  TRUE
## kurtosis_yaw_forearm       46.532872    0.01455922   FALSE  TRUE
## skewness_roll_forearm     206.892308    1.64519182   FALSE  TRUE
## skewness_pitch_forearm    200.716418    1.62335299   FALSE  TRUE
## skewness_yaw_forearm       46.532872    0.01455922   FALSE  TRUE
## max_roll_forearm           21.666667    1.40496469   FALSE  TRUE
## max_picth_forearm           2.708333    0.88083279   FALSE FALSE
## max_yaw_forearm           203.757576    0.28390478   FALSE  TRUE
## min_roll_forearm           21.666667    1.46320157   FALSE  TRUE
## min_pitch_forearm           4.062500    0.98274732   FALSE FALSE
## min_yaw_forearm           203.757576    0.28390478   FALSE  TRUE
## amplitude_roll_forearm     21.666667    1.53599767   FALSE  TRUE
## amplitude_pitch_forearm     4.785714    1.01914537   FALSE FALSE
## amplitude_yaw_forearm      60.304933    0.02183883   FALSE  TRUE
## total_accel_forearm         1.088664    0.50229308   FALSE FALSE
## var_accel_forearm           3.000000    2.08924802   FALSE FALSE
## avg_roll_forearm           32.500000    1.63063260   FALSE  TRUE
## stddev_roll_forearm        68.000000    1.61607338   FALSE  TRUE
## var_roll_forearm           68.000000    1.61607338   FALSE  TRUE
## avg_pitch_forearm          65.000000    1.63791221   FALSE  TRUE
## stddev_pitch_forearm       65.000000    1.63791221   FALSE  TRUE
## var_pitch_forearm          65.000000    1.63791221   FALSE  TRUE
## avg_yaw_forearm            65.000000    1.63791221   FALSE  TRUE
## stddev_yaw_forearm         67.000000    1.62335299   FALSE  TRUE
## var_yaw_forearm            67.000000    1.62335299   FALSE  TRUE
## gyros_forearm_x             1.074792    2.06012958   FALSE FALSE
## gyros_forearm_y             1.029412    5.24859868   FALSE FALSE
## gyros_forearm_z             1.137313    2.14020528   FALSE FALSE
## accel_forearm_x             1.047619    5.67809565   FALSE FALSE
## accel_forearm_y             1.000000    7.11217879   FALSE FALSE
## accel_forearm_z             1.185185    4.04746306   FALSE FALSE
## magnet_forearm_x            1.000000   10.61367111   FALSE FALSE
## magnet_forearm_y            1.163934   13.37264323   FALSE FALSE
## magnet_forearm_z            1.000000   11.84392517   FALSE FALSE
## classe                      1.469526    0.03639805   FALSE FALSE
```

c. NA Values

From the summary, it is very evident that most of the variables are composed on NA values. If large portion of the covariate is just NA values. It is might as well good to consider removing this covariates.

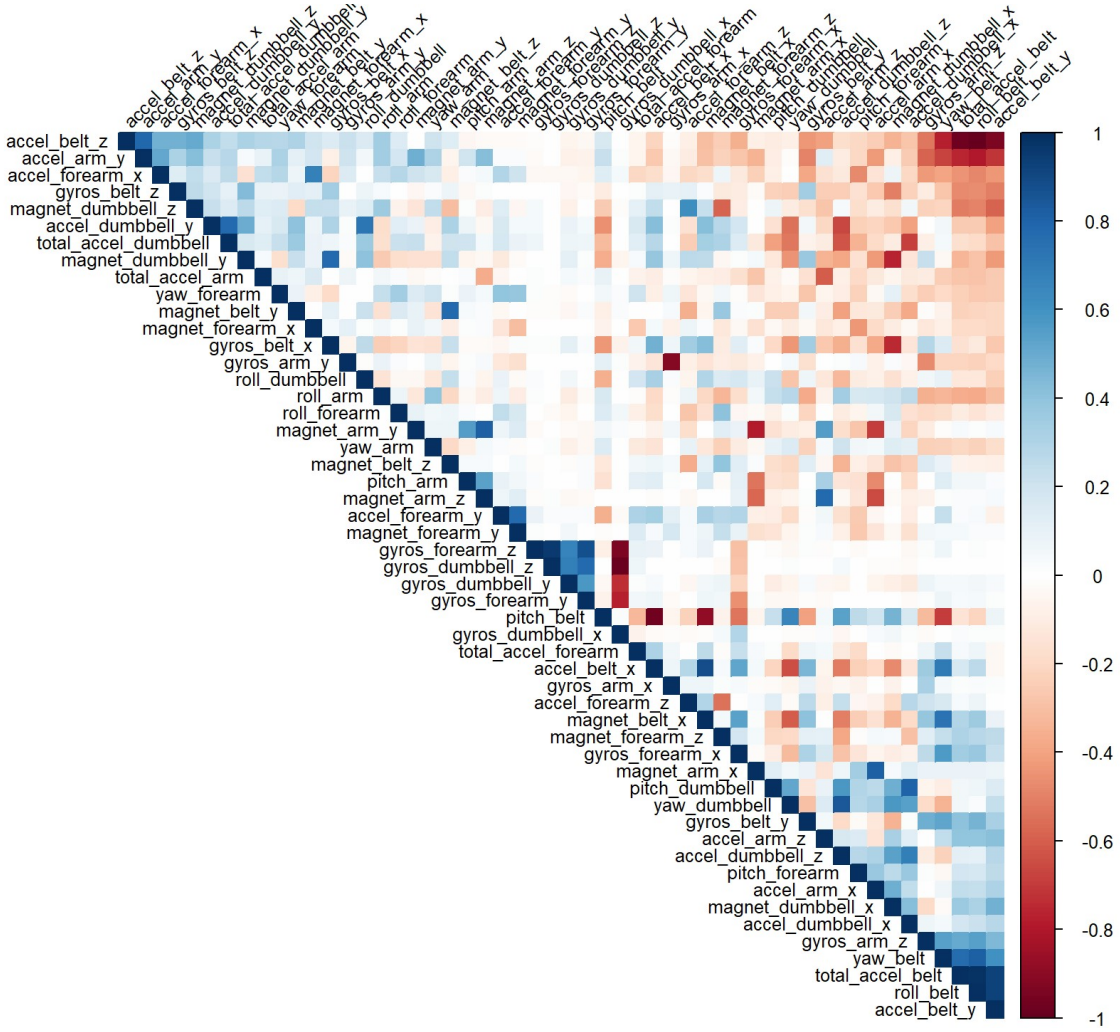
```
allNA <- sapply(train_set_clean, function(x) mean(is.na(x))) > 0.95
train_set_clean <- train_set_clean[, allNA==FALSE]
test_set_clean <- test_set_clean[, allNA==FALSE]
```

D. Correlation Analysis

Lastly, correlation analysis is applied to the partly cleaned data. The goal is to eliminate highly correlated covariates because from the lesson, it is highly emphasized that highly correlated variables don't improve models for the reason that it mask interactions between different features.

In order to visualize the correlation of each covariates, here is the correlation plot.

```
corrplot(cor(train_set_clean[, -53]), order = "FPC", method = "color", type = "upper", tl.cex = 0.8, tl.col
= rgb(0, 0, 0), tl.srt=45)
```



As can be noticed, some of the covariates are highly correlated. For this purpose, highly correlated covariates are defined to have a cut off of at least 0.90 in absolute value. Identified variables will then be excluded from the predictors.

```
c <- findCorrelation(abs(cor(train_set_clean[, -53])), cutoff = .90)
train_set_clean <- train_set_clean[, -c]
test_set_clean <- test_set_clean[, -c]
dim(train_set_clean)
```

```
## [1] 13737    46
```

```
dim(test_set_clean)
```

```
## [1] 5885   46
```

There are a total of seven highly correlated variables based on the threshold. After all the cleaning process applied to the original partitioned data set, the number of covariates for the modeling has been reduced from **159 predictors** to only **45 predictors** plus **one outcome variable**.

III. Prediction Model Building

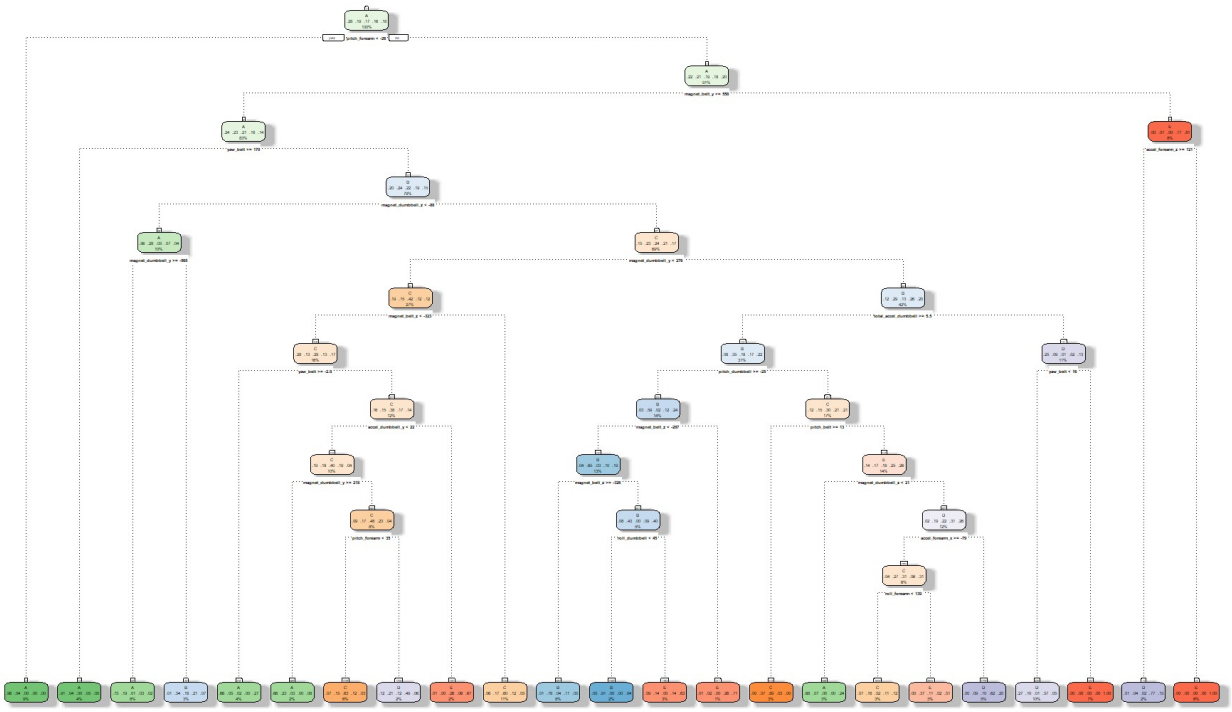
For this project, there will be three algorithm to be used in order to discover the best model to predict the class or fashion of performing the Unilateral Dumbbell Biceps Curl based on the given variables. The three algorithms are:

a. Decision Tree
b. Random Forests
c. Gradient Boosting Method

A. Decision Tree Algorithm

A Decision Tree is a supervised learning predictive model that uses a set of binary rules to calculate a target value.
Source: *A Guide to Machine Learning in R for Beginners: Decision Trees (https://medium.com/analytics-vidhya/a-guide-to-machine-learning-in-r-for-beginners-decision-trees-c24dfd490abb)*

```
set.seed(123456789)
model_decisiontree <- rpart(classe ~ ., data=train_set_clean, method="class")
fancyRpartPlot(model_decisiontree)
```



Rattle 2019-Jun-18 23:42:38 10012211

```
prediction_model_decisiontree <- predict(model_decisiontree, newdata=test_set_clean, type="class")
cm_model_decisiontree <- confusionMatrix(prediction_model_decisiontree, test_set_clean$classe)
cm_model_decisiontree
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1387  184   19   28  133
##          B   18  546   46   90   49
##          C   79  191  853  155   71
##          D  178  138   59  643   98
##          E   12   80   49   48  731
##
## Overall Statistics
##
##                Accuracy : 0.7069
##                  95% CI : (0.6951, 0.7185)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6294
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8286  0.47937   0.8314   0.6670   0.6756
## Specificity            0.9136  0.95723   0.8979   0.9039   0.9606
## Pos Pred Value         0.7921  0.72897   0.6323   0.5762   0.7946
## Neg Pred Value         0.9306  0.88454   0.9619   0.9327   0.9293
## Prevalence             0.2845  0.19354   0.1743   0.1638   0.1839
## Detection Rate         0.2357  0.09278   0.1449   0.1093   0.1242
## Detection Prevalence   0.2975  0.12727   0.2292   0.1896   0.1563
## Balanced Accuracy      0.8711  0.71830   0.8647   0.7854   0.8181
```

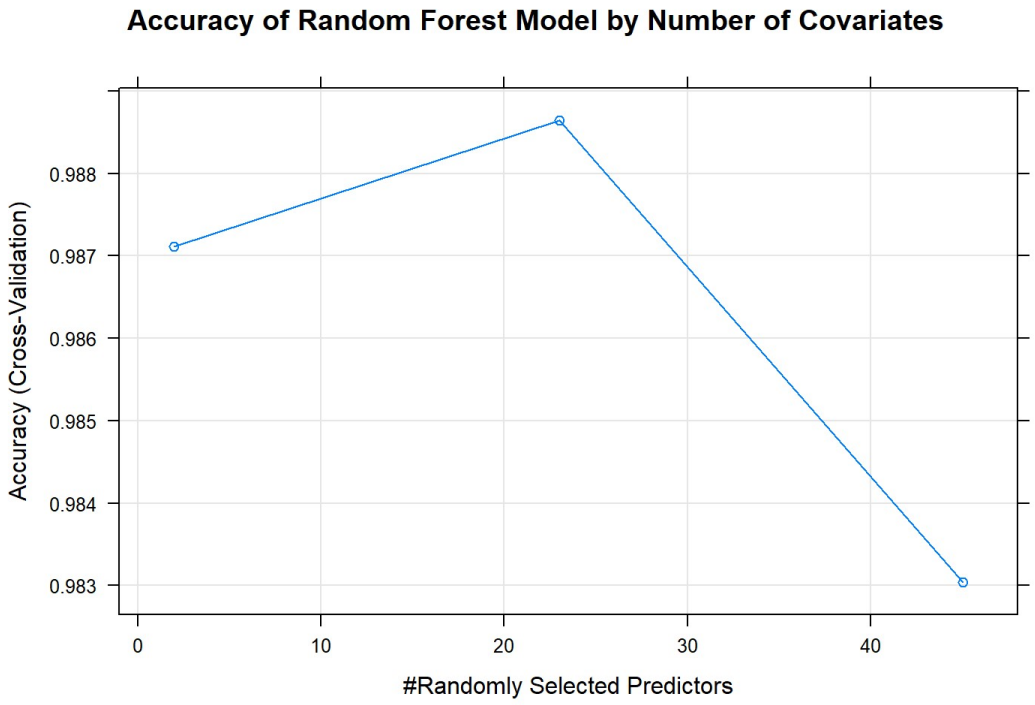
B. Random Forest Algorithm

In Random Forests the idea is to decorrelate the several trees which are generated by the different bootstrapped samples from training Data. And then we simply reduce the Variance in the Trees by averaging them.
Source: *Random Forests in R (https://datascienceplus.com/random-forests-in-r/)*

```
set.seed(123456789)
traincontrol_ranfor <- trainControl(method="cv", number=3, verboseIter=FALSE)
model_randomforest <- train(classe ~ ., data=train_set_clean, method="rf", trControl=traincontrol_ranfor)
model_randomforest
```

```
## Random Forest
##
## 13737 samples
##    45 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (3 fold)
## Summary of sample sizes: 9160, 9157, 9157
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa
##    2    0.9871156  0.9837005
##   23    0.9886443  0.9856364
##   45    0.9830392  0.9785436
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 23.
```

```
plot(model_randomforest,main="Accuracy of Random Forest Model by Number of Covariates")
```




```
prediction_model_ranfor <- predict(model_randomforest, newdata=test_set_clean)
cm_model_ranfor<- confusionMatrix(prediction_model_ranfor, test_set_clean$classe)
cm_model_ranfor
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673   10    0    0    0
##          B    1 1128   13    0    0
##          C    0    1 1010   15    2
##          D    0    0    3  948    1
##          E    0    0    0    1 1079
##
## Overall Statistics
##
##                Accuracy : 0.992
##                  95% CI : (0.9894, 0.9941)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9899
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9903   0.9844   0.9834   0.9972
## Specificity            0.9976   0.9971   0.9963   0.9992   0.9998
## Pos Pred Value         0.9941   0.9877   0.9825   0.9958   0.9991
## Neg Pred Value         0.9998   0.9977   0.9967   0.9968   0.9994
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1917   0.1716   0.1611   0.1833
## Detection Prevalence   0.2860   0.1941   0.1747   0.1618   0.1835
## Balanced Accuracy      0.9985   0.9937   0.9904   0.9913   0.9985
```

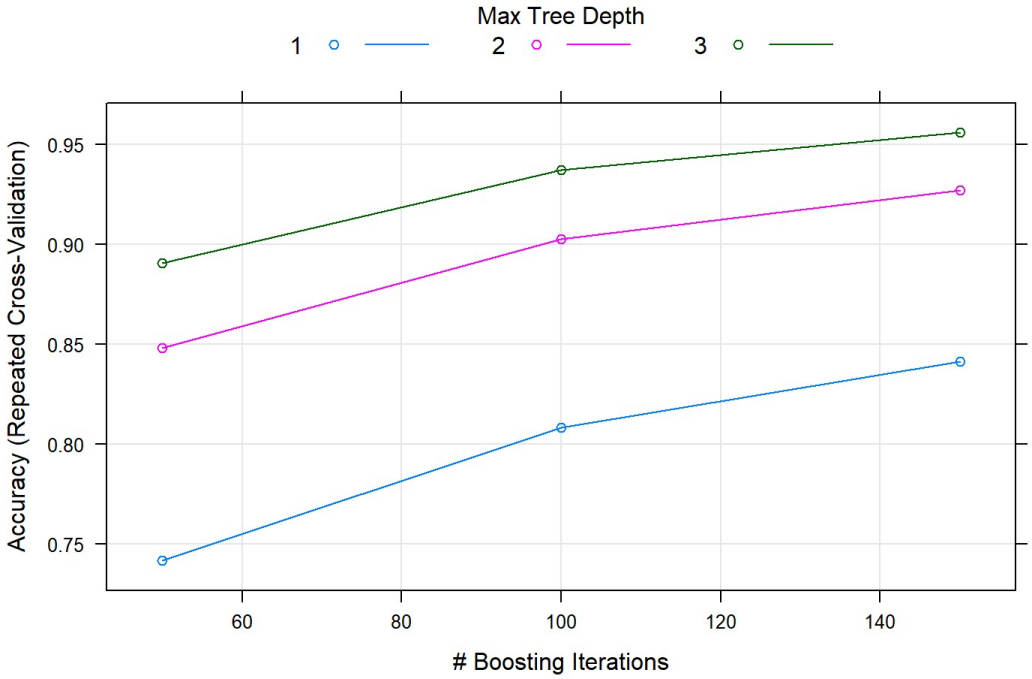
C. Gradient Boosting Method

The main idea of boosting is to add new models to the ensemble sequentially. At each particular iteration, a new weak, base-learner model is trained with respect to the error of the whole ensemble learnt so far.
Source: *Gradient Boosting Machines (http://uc-r.github.io/gbm_regression)*

```
set.seed(123456789)
traincontrol_gbm <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
model_gbm  <- train(classe ~ ., data=train_set_clean, method = "gbm", trControl = traincontrol_gbm, verbos
e = FALSE)
model_gbm
```

```
## Stochastic Gradient Boosting
##
## 13737 samples
##    45 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 1 times)
## Summary of sample sizes: 10989, 10988, 10990, 10990, 10991
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7417204  0.6728296
##   1                  100      0.8083285  0.7574831
##   1                  150      0.8415238  0.7994574
##   2                   50      0.8480031  0.8074418
##   2                  100      0.9027450  0.8769248
##   2                  150      0.9269861  0.9075945
##   3                   50      0.8905879  0.8614977
##   3                  100      0.9370308  0.9203304
##   3                  150      0.9558125  0.9440984
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.1
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were n.trees = 150,
##  interaction.depth = 3, shrinkage = 0.1 and n.minobsinnode = 10.
```

```
plot(model_gbm)
```




```
prediction_model_gbm <- predict(model_gbm, newdata=test_set_clean)
cm_model_gbm <- confusionMatrix(prediction_model_gbm, test_set_clean$classe)
cm_model_gbm
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1644   46    0    1    5
##          B   18 1052   31    7   10
##          C    9   36  979   35   13
##          D    2    2   16  915    6
##          E    1    3    0    6 1048
##
## Overall Statistics
##
##                Accuracy : 0.958
##                  95% CI : (0.9526, 0.963)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9469
##
##  Mcnemar's Test P-Value : 1.571e-07
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9821   0.9236   0.9542   0.9492   0.9686
## Specificity            0.9877   0.9861   0.9809   0.9947   0.9979
## Pos Pred Value         0.9693   0.9410   0.9132   0.9724   0.9905
## Neg Pred Value         0.9928   0.9817   0.9902   0.9901   0.9930
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2794   0.1788   0.1664   0.1555   0.1781
## Detection Prevalence   0.2882   0.1900   0.1822   0.1599   0.1798
## Balanced Accuracy      0.9849   0.9549   0.9675   0.9719   0.9832
```

IV. Result Summary and Conclusion

Presented below is the table to summarize the output characteristics of the model created using the different algorithms.

Algorithm	**Accuracy**	**Kappa**	**95% CI**
Decision Tree	70.69%	62.94%	69.51% - 71.85%
Random Forest	99.20%	98.99%	98.94% - 99.41%
Gradient Boosting Method	95.82%	94.71%	95.28% - 96.32%

From the result above, it is clear that **Random Forest Algorithm** provided the best predictive model for the class or fashion of performing the Unilateral Dumbbell Biceps Curl based on the given variables.

V. Application

This section shows the application of the selected best predictive model (using Random Forest Algorithm) to the given set of testing data for the evaluation exercises.

```
evaluation_prediction <- predict(model_randomforest, newdata=pml_validation)
evaluation_prediction
```

```
##  [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
```