Course Project - Practical Machine Learning

Alejandro, Julian Paolo S.

August 5, 2019

I. Overview

A. Background

Devices such as Jawbone Up, Nike FuelBand, and Fitbit make it 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¢Â□Â□ 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.

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  1 2 3 4 5 6 7 11 13 14 ...
##  $ 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 1323084232 1323084232 1323084232 1323084232 ...
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 304277 368296 500302 560359 576390 ...
##  $ 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 1.45 1.42 1.42 ...
##  $ pitch_belt              : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.18 8.2 8.21 ...
##  $ 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 0.02 0.02 0.02 0.02 0.03 0.02 0.02 ...
##  $ 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 -0.02 ...
##  $ accel_belt_x            : int  -21 -22 -20 -22 -21 -21 -22 -21 -22 -22 ...
##  $ accel_belt_y            : int  4 4 5 3 2 4 3 2 4 4 ...
##  $ accel_belt_z            : int  22 22 23 21 24 21 21 23 21 21 ...
##  $ magnet_belt_x           : int  -3 -7 -2 -6 -6 0 -4 -5 -3 -8 ...
##  $ magnet_belt_y           : int  599 608 600 604 600 603 599 596 606 598 ...
##  $ magnet_belt_z           : int  -313 -311 -305 -310 -302 -312 -311 -317 -309 -310 ...
##  $ 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 21.5 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 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 0 0 -0.02 -0.03 ...
##  $ accel_arm_x            : int  -288 -290 -289 -289 -289 -289 -289 -290 -287 -288 ...
##  $ accel_arm_y            : int  109 110 110 111 111 111 111 110 111 111 ...
##  $ accel_arm_z            : int  -123 -125 -126 -123 -123 -122 -125 -123 -124 -124 ...
##  $ magnet_arm_x           : int  -368 -369 -368 -372 -374 -369 -373 -366 -372 -371 ...
##  $ magnet_arm_y           : int  337 337 344 344 337 342 336 339 338 331 ...
##  $ magnet_arm_z           : int  516 513 513 512 506 513 509 509 509 523 ...
##  $ 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  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  : 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                 1.116883    7.97845235   FALSE FALSE
## pitch_belt                1.206612   12.28070175   FALSE FALSE
## yaw_belt                  1.093664   12.87762976   FALSE FALSE
## total_accel_belt          1.077580    0.20382907   FALSE FALSE
## kurtosis_roll_belt     1344.000000    2.10380724   FALSE  TRUE
## kurtosis_picth_belt     560.000000    1.79078401   FALSE  TRUE
## kurtosis_yaw_belt        45.252525    0.01455922   FALSE  TRUE
## skewness_roll_belt     1493.333333    2.10380724   FALSE  TRUE
## skewness_roll_belt.1    560.000000    1.87085972   FALSE  TRUE
## skewness_yaw_belt        45.252525    0.01455922   FALSE  TRUE
## max_roll_belt             1.222222    1.20113562   FALSE FALSE
## max_picth_belt            2.051282    0.14559220   FALSE FALSE
## max_yaw_belt            610.909091    0.42949698   FALSE  TRUE
## min_roll_belt             1.000000    1.11378030   FALSE FALSE
## min_pitch_belt            2.540541    0.10191454   FALSE FALSE
## min_yaw_belt            610.909091    0.42949698   FALSE  TRUE
## amplitude_roll_belt       1.260870    0.88083279   FALSE FALSE
## amplitude_pitch_belt      2.867925    0.09463493   FALSE FALSE
## amplitude_yaw_belt       48.000000    0.02911844   FALSE  TRUE
## var_total_accel_belt      1.354839    0.39309893   FALSE FALSE
## avg_roll_belt             1.083333    1.16473757   FALSE FALSE
## stddev_roll_belt          1.051282    0.42949698   FALSE FALSE
## var_roll_belt             1.500000    0.54597074   FALSE FALSE
## avg_pitch_belt            1.142857    1.28849094   FALSE FALSE
## stddev_pitch_belt         1.100000    0.26206595   FALSE FALSE
## var_pitch_belt            1.106667    0.37126010   FALSE FALSE
## avg_yaw_belt              1.250000    1.42680352   FALSE FALSE
## stddev_yaw_belt           1.676471    0.35670088   FALSE FALSE
## var_yaw_belt              1.777778    0.90267162   FALSE FALSE
## gyros_belt_x              1.049163    0.92451045   FALSE FALSE
## gyros_belt_y              1.160083    0.47317464   FALSE FALSE
## gyros_belt_z              1.063149    1.15745796   FALSE FALSE
## accel_belt_x              1.052441    1.17201718   FALSE FALSE
## accel_belt_y              1.093151    0.99002693   FALSE FALSE
## accel_belt_z              1.084437    2.12564607   FALSE FALSE
## magnet_belt_x             1.007663    2.22028099   FALSE FALSE
## magnet_belt_y             1.115556    2.11836646   FALSE FALSE
## magnet_belt_z             1.058462    3.20302832   FALSE FALSE
## roll_arm                 52.434783   17.46378394   FALSE FALSE
## pitch_arm                86.178571   20.06260464   FALSE FALSE
## yaw_arm                  32.160000   19.22544952   FALSE FALSE
## total_accel_arm           1.015974    0.48045425   FALSE FALSE
## var_accel_arm             4.000000    2.10380724   FALSE FALSE
## avg_roll_arm             51.000000    1.79806362   FALSE  TRUE
## stddev_roll_arm          51.000000    1.79806362   FALSE  TRUE
## var_roll_arm             51.000000    1.79806362   FALSE  TRUE
## avg_pitch_arm            51.000000    1.79806362   FALSE  TRUE
## stddev_pitch_arm         51.000000    1.79806362   FALSE  TRUE
## var_pitch_arm            51.000000    1.79806362   FALSE  TRUE
## avg_yaw_arm              51.000000    1.79806362   FALSE  TRUE
## stddev_yaw_arm           53.000000    1.78350440   FALSE  TRUE
## var_yaw_arm              53.000000    1.78350440   FALSE  TRUE
## gyros_arm_x               1.000000    4.50607847   FALSE FALSE
## gyros_arm_y               1.392573    2.68617602   FALSE FALSE
## gyros_arm_z               1.150273    1.71070831   FALSE FALSE
```

```
## accel_arm_x                1.173554    5.61257917   FALSE FALSE
## accel_arm_y                1.056604    3.82907476   FALSE FALSE
## accel_arm_z                1.098901    5.59801995   FALSE FALSE
## magnet_arm_x               1.050000    9.65276261   FALSE FALSE
## magnet_arm_y               1.046154    6.26046444   FALSE FALSE
## magnet_arm_z               1.103896    9.14318993   FALSE FALSE
## kurtosis_roll_arm        258.461538    1.79806362   FALSE  TRUE
## kurtosis_picth_arm       253.584906    1.79078401   FALSE  TRUE
## kurtosis_yaw_arm        1680.000000    2.10380724   FALSE  TRUE
## skewness_roll_arm        263.529412    1.80534323   FALSE  TRUE
## skewness_pitch_arm       253.584906    1.79078401   FALSE  TRUE
## skewness_yaw_arm        1680.000000    2.11108685   FALSE  TRUE
## max_roll_arm              17.000000    1.57967533   FALSE FALSE
## max_picth_arm             17.000000    1.50687923   FALSE FALSE
## max_yaw_arm                1.333333    0.34214166   FALSE FALSE
## min_roll_arm              17.000000    1.57967533   FALSE FALSE
## min_pitch_arm             17.000000    1.63791221   FALSE FALSE
## min_yaw_arm                1.052632    0.26206595   FALSE FALSE
## amplitude_roll_arm        25.500000    1.70342870   FALSE  TRUE
## amplitude_pitch_arm       17.666667    1.65247143   FALSE FALSE
## amplitude_yaw_arm          1.263158    0.35670088   FALSE FALSE
## roll_dumbbell              1.010526   86.68559365   FALSE FALSE
## pitch_dumbbell             2.294737   84.62546408   FALSE FALSE
## yaw_dumbbell               1.172840   86.11778409   FALSE FALSE
## kurtosis_roll_dumbbell   4480.000000    2.12564607   FALSE  TRUE
## kurtosis_picth_dumbbell  6720.000000    2.13292568   FALSE  TRUE
## kurtosis_yaw_dumbbell      45.252525    0.01455922   FALSE  TRUE
## skewness_roll_dumbbell   6720.000000    2.14748489   FALSE  TRUE
## skewness_pitch_dumbbell  6720.000000    2.15476450   FALSE  TRUE
## skewness_yaw_dumbbell      45.252525    0.01455922   FALSE  TRUE
## max_roll_dumbbell           1.000000    1.90725777   FALSE FALSE
## max_picth_dumbbell          1.000000    1.87085972   FALSE FALSE
## max_yaw_dumbbell          840.000000    0.45861542   FALSE  TRUE
## min_roll_dumbbell           1.000000    1.80534323   FALSE FALSE
## min_pitch_dumbbell          1.666667    1.92909660   FALSE FALSE
## min_yaw_dumbbell          840.000000    0.45861542   FALSE  TRUE
## amplitude_roll_dumbbell     6.000000    2.07468880   FALSE FALSE
## amplitude_pitch_dumbbell    6.000000    2.05284997   FALSE FALSE
## amplitude_yaw_dumbbell     45.714286    0.02183883   FALSE  TRUE
## total_accel_dumbbell        1.050515    0.30574361   FALSE FALSE
## var_accel_dumbbell         14.000000    2.06740919   FALSE FALSE
## avg_roll_dumbbell           1.000000    2.12564607   FALSE FALSE
## stddev_roll_dumbbell       12.000000    2.08196841   FALSE FALSE
## var_roll_dumbbell          12.000000    2.08196841   FALSE FALSE
## avg_pitch_dumbbell          1.000000    2.12564607   FALSE FALSE
## stddev_pitch_dumbbell      12.000000    2.08196841   FALSE FALSE
## var_pitch_dumbbell         12.000000    2.08196841   FALSE FALSE
## avg_yaw_dumbbell            1.000000    2.12564607   FALSE FALSE
## stddev_yaw_dumbbell        12.000000    2.08196841   FALSE FALSE
## var_yaw_dumbbell           12.000000    2.08196841   FALSE FALSE
## gyros_dumbbell_x            1.044084    1.68158987   FALSE FALSE
## gyros_dumbbell_y            1.281174    1.96549465   FALSE FALSE
## gyros_dumbbell_z            1.037209    1.41952391   FALSE FALSE
## accel_dumbbell_x            1.029536    2.97736041   FALSE FALSE
## accel_dumbbell_y            1.142857    3.29766325   FALSE FALSE
## accel_dumbbell_z            1.011561    2.91184393   FALSE FALSE
## magnet_dumbbell_x           1.016807    7.77462328   FALSE FALSE
```

```
## magnet_dumbbell_y          1.283186    5.99111888   FALSE FALSE
## magnet_dumbbell_z          1.092308    4.84094053   FALSE FALSE
## roll_forearm              11.666667   13.62742957   FALSE FALSE
## pitch_forearm             62.674419   19.18905147   FALSE FALSE
## yaw_forearm               14.966667   12.89218898   FALSE FALSE
## kurtosis_roll_forearm    184.109589    1.63791221   FALSE  TRUE
## kurtosis_picth_forearm   181.621622    1.63791221   FALSE  TRUE
## kurtosis_yaw_forearm      45.252525    0.01455922   FALSE  TRUE
## skewness_roll_forearm    186.666667    1.65247143   FALSE  TRUE
## skewness_pitch_forearm   181.621622    1.60879377   FALSE  TRUE
## skewness_yaw_forearm      45.252525    0.01455922   FALSE  TRUE
## max_roll_forearm          24.000000    1.44864235   FALSE  TRUE
## max_picth_forearm          4.500000    0.90995123   FALSE FALSE
## max_yaw_forearm          184.109589    0.29846400   FALSE  TRUE
## min_roll_forearm          24.000000    1.46320157   FALSE  TRUE
## min_pitch_forearm          3.272727    0.93179006   FALSE FALSE
## min_yaw_forearm          184.109589    0.29846400   FALSE  TRUE
## amplitude_roll_forearm    24.000000    1.54327728   FALSE  TRUE
## amplitude_pitch_forearm    4.352941    1.02642498   FALSE FALSE
## amplitude_yaw_forearm     60.000000    0.02183883   FALSE  TRUE
## total_accel_forearm        1.125428    0.49501347   FALSE FALSE
## var_accel_forearm          4.000000    2.14020528   FALSE FALSE
## avg_roll_forearm          72.000000    1.64519182   FALSE  TRUE
## stddev_roll_forearm       74.000000    1.63063260   FALSE  TRUE
## var_roll_forearm          74.000000    1.63063260   FALSE  TRUE
## avg_pitch_forearm         72.000000    1.64519182   FALSE  TRUE
## stddev_pitch_forearm      72.000000    1.64519182   FALSE  TRUE
## var_pitch_forearm         72.000000    1.64519182   FALSE  TRUE
## avg_yaw_forearm           72.000000    1.64519182   FALSE  TRUE
## stddev_yaw_forearm        74.000000    1.63063260   FALSE  TRUE
## var_yaw_forearm           74.000000    1.63063260   FALSE  TRUE
## gyros_forearm_x            1.107438    2.03829075   FALSE FALSE
## gyros_forearm_y            1.095420    5.19764141   FALSE FALSE
## gyros_forearm_z            1.104348    2.11836646   FALSE FALSE
## accel_forearm_x            1.030769    5.68537526   FALSE FALSE
## accel_forearm_y            1.171429    7.11945840   FALSE FALSE
## accel_forearm_z            1.035714    4.10569993   FALSE FALSE
## magnet_forearm_x           1.050000   10.59183228   FALSE FALSE
## magnet_forearm_y           1.310345   13.25616947   FALSE FALSE
## magnet_forearm_z           1.023256   11.70561258   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.

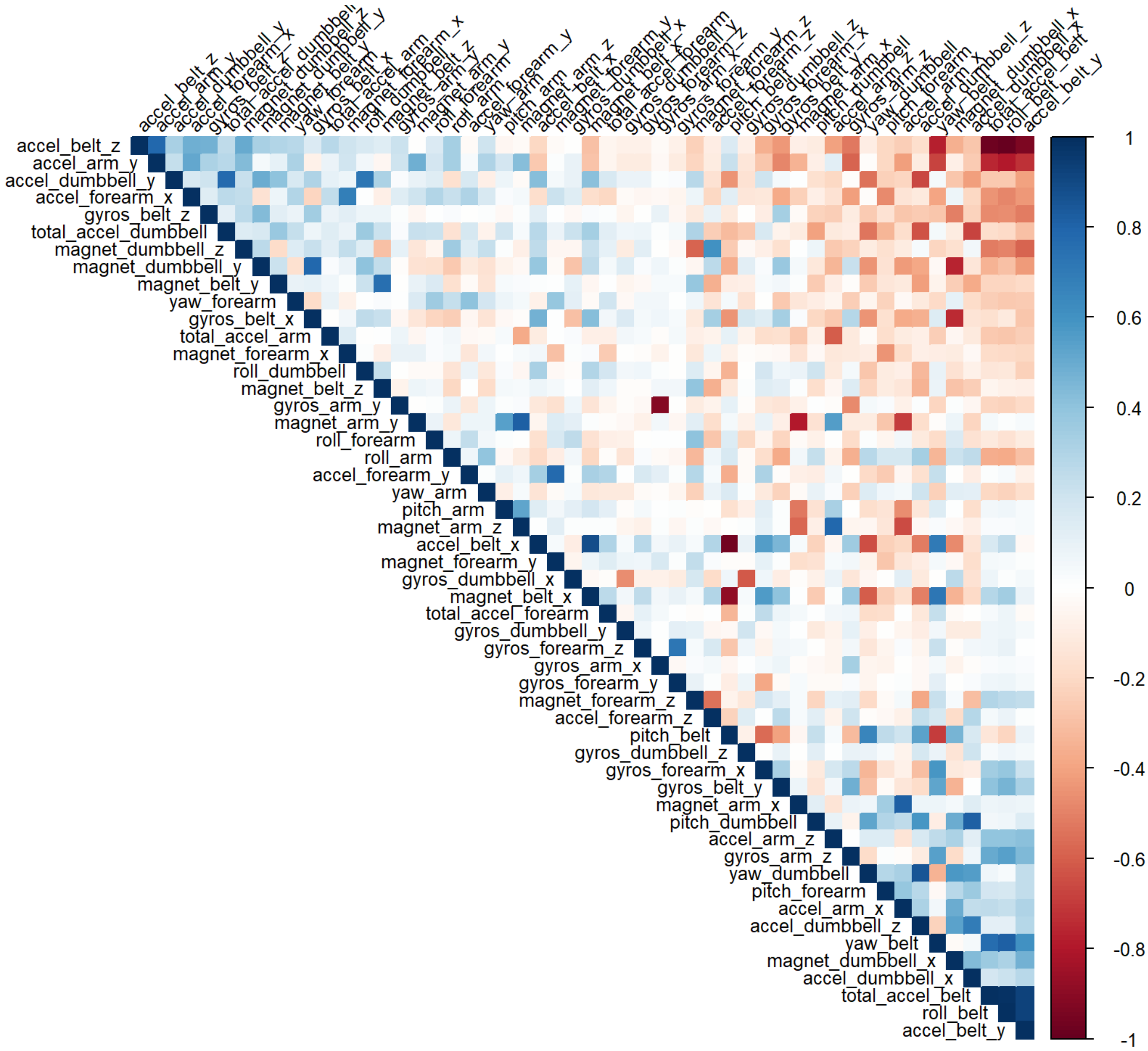
```
allNA <- sapply(train_set_clean, function(x) mean(is.na(x))) > 0.95
train_set_clean <- train_set_clean[, allNA==FALSE]
test_set_clean <- test_set_clean[, allNA==FALSE]
```

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, t
l.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    48
```

```
dim(test_set_clean)
```

```
## [1] 5885   48
```

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

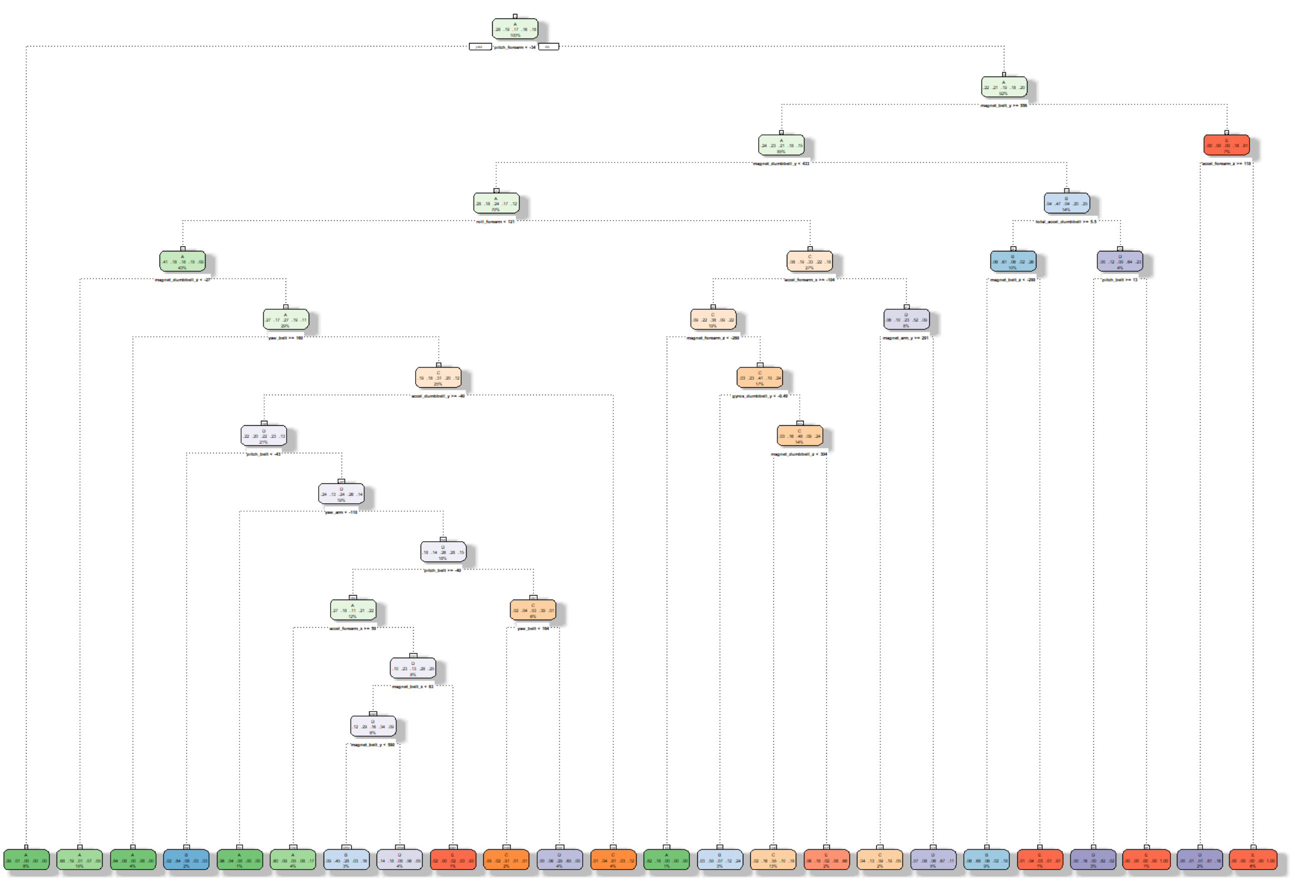
For this project, there will be three algorithm to be used in order to discover the best model to predict the class or fashion of performing the Unilateral Dumbbell Biceps Curl based on the given variables. The three algorithms are:

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-Aug-05 08:13:47 10012190

```
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 1544  233   26   77   85
##          B   43  619  108   43  180
##          C   33  169  777  102  150
##          D   41  101  107  723   67
##          E   13   17    8   19  600
## 
## Overall Statistics
## 
##                Accuracy : 0.7244
##                  95% CI : (0.7128, 0.7358)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.6495
## 
##  Mcnemar's Test P-Value : < 2.2e-16
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9223   0.5435   0.7573   0.7500   0.5545
## Specificity            0.9000   0.9212   0.9066   0.9358   0.9881
## Pos Pred Value         0.7858   0.6234   0.6312   0.6959   0.9132
## Neg Pred Value         0.9668   0.8937   0.9465   0.9503   0.9078
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2624   0.1052   0.1320   0.1229   0.1020
## Detection Prevalence   0.3339   0.1687   0.2092   0.1766   0.1116
## Balanced Accuracy      0.9112   0.7323   0.8319   0.8429   0.7713
```

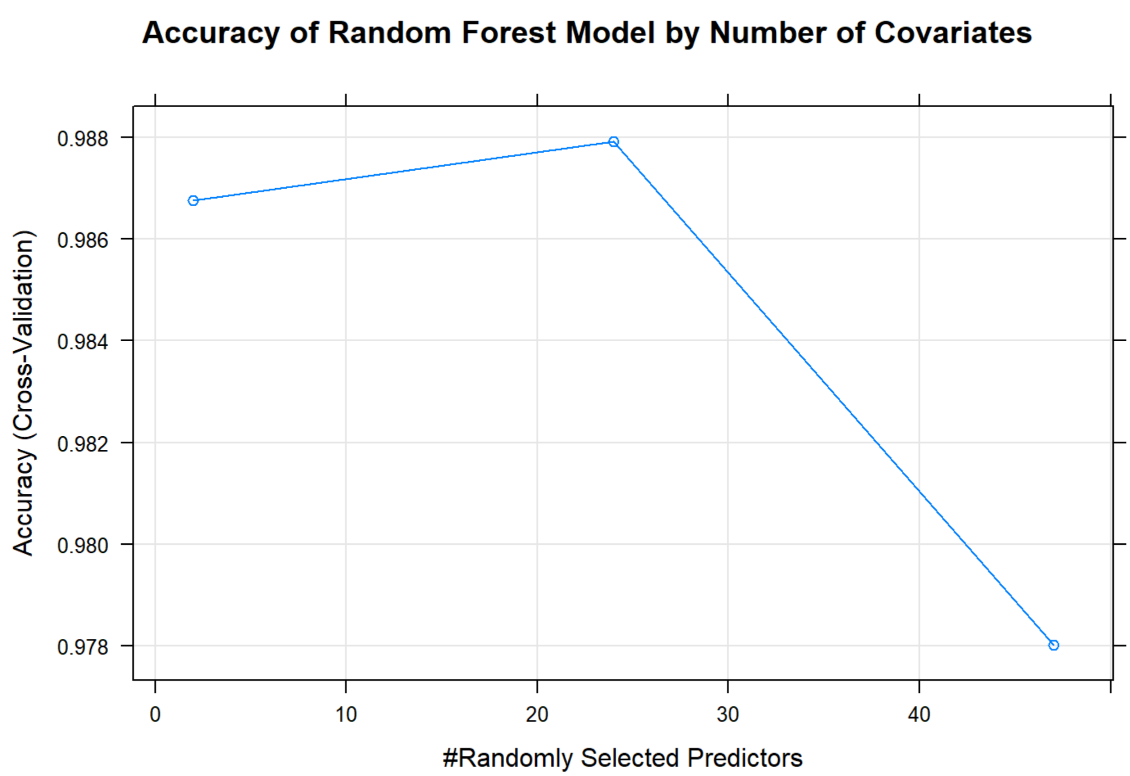
B. Random Forest Algorithm

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_ranfo
r)
model_randomforest
```

```
## Random Forest
##
## 13737 samples
##    47 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    0.9867517  0.9832381
##   24    0.9879161  0.9847120
##   47    0.9780161  0.9721876
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was mtry = 24.
```

```
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    5    0    0    0
##          B    0 1127    4    0    0
##          C    0    7 1020   10    2
##          D    0    0    2  954    2
##          E    1    0    0    0 1078
## 
## Overall Statistics
## 
##                Accuracy : 0.9944
##                  95% CI : (0.9921, 0.9961)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9929
## 
##  Mcnemar's Test P-Value : NA
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9994   0.9895   0.9942   0.9896   0.9963
## Specificity            0.9988   0.9992   0.9961   0.9992   0.9998
## Pos Pred Value         0.9970   0.9965   0.9817   0.9958   0.9991
## Neg Pred Value         0.9998   0.9975   0.9988   0.9980   0.9992
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2843   0.1915   0.1733   0.1621   0.1832
## Detection Prevalence   0.2851   0.1922   0.1766   0.1628   0.1833
## Balanced Accuracy      0.9991   0.9943   0.9951   0.9944   0.9980
```

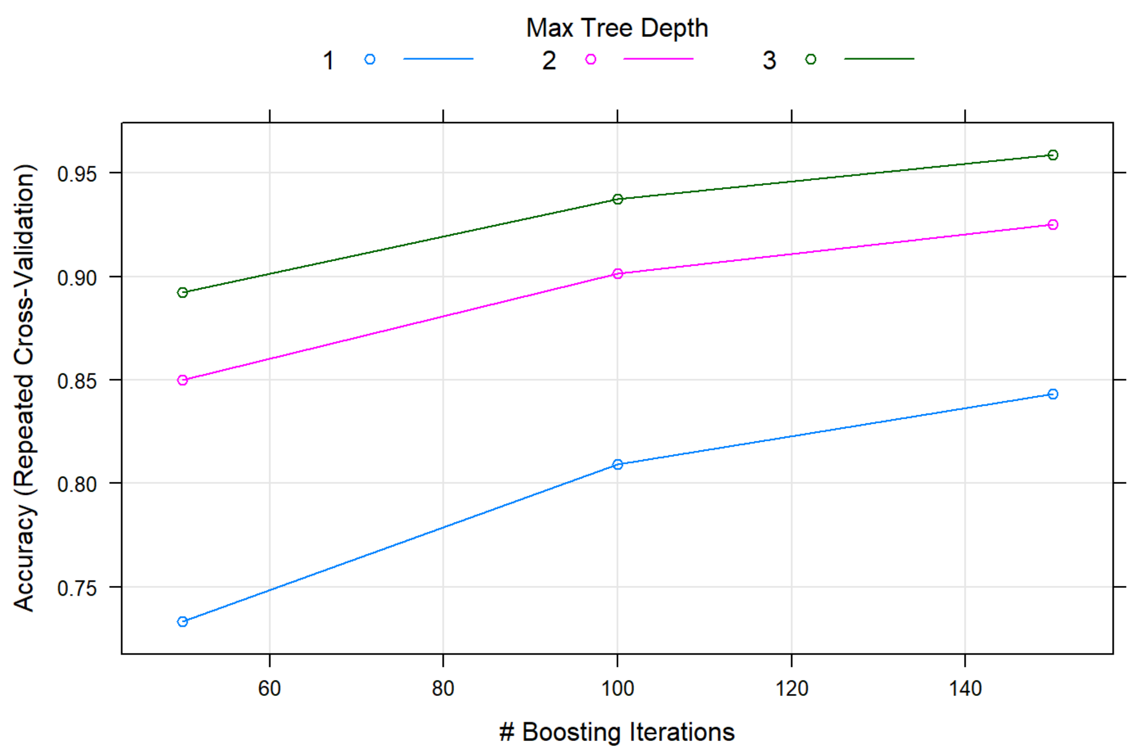
C. Gradient Boosting Method

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(1)
traincontrol_gbm <- trainControl(method = "repeatedcv", number = 5, repeats = 1)
model_gbm  <- train(classe ~ ., data=train_set_clean, method = "gbm", trControl = traincontrol_gbm, ver
bose = FALSE)
model_gbm
```

```
## Stochastic Gradient Boosting
##
## 13737 samples
##    47 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold, repeated 1 times)
## Summary of sample sizes: 10991, 10988, 10991, 10988, 10990
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  Accuracy   Kappa
##   1                   50      0.7332008  0.6617833
##   1                  100      0.8091285  0.7584652
##   1                  150      0.8431971  0.8015465
##   2                   50      0.8498201  0.8097262
##   2                  100      0.9015060  0.8753519
##   2                  150      0.9252363  0.9054094
##   3                   50      0.8921867  0.8634879
##   3                  100      0.9372478  0.9205898
##   3                  150      0.9585778  0.9475880
##
## 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 1656   24    0    2    1
##          B   16 1076   26    1   11
##          C    1   36  985   32    8
##          D    1    3   14  924   16
##          E    0    0    1    5 1046
## 
## Overall Statistics
## 
##                Accuracy : 0.9664
##                  95% CI : (0.9614, 0.9708)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
## 
##                   Kappa : 0.9574
## 
##  Mcnemar's Test P-Value : 9.125e-05
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9892   0.9447   0.9600   0.9585   0.9667
## Specificity            0.9936   0.9886   0.9842   0.9931   0.9988
## Pos Pred Value         0.9840   0.9522   0.9275   0.9645   0.9943
## Neg Pred Value         0.9957   0.9868   0.9915   0.9919   0.9926
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2814   0.1828   0.1674   0.1570   0.1777
## Detection Prevalence   0.2860   0.1920   0.1805   0.1628   0.1788
## Balanced Accuracy      0.9914   0.9667   0.9721   0.9758   0.9827
```

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
```