Project Assignment

Title: Predicting the Quality of Weight Lifting Activity

Sergio Vicente Simioni

July, 18, 2015

Report Content

a. **Executive Summary**
b. **Conclusions/Questions addressing**
c. **Data Analysis**

Content Description

Executive Summary 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 - 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. The data was collected 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 ("classe"):
A - Regular
B - Throwing the elbows to the front
C - Lifting the dumbbell only halfway
D - Lowering the dumbbell only halfway
E - Throwing the hips to the front

The goal of this project is to predict the manner in which the participants did the exercise. This is the "classe" variable in the training set. You may use any of the other variables to predict with. You should create a report describing how you built your model, how you used cross validation, what you think the expected out of sample error is, and why you made the choices you did. You will also use your prediction model to predict 20 different test cases.

The training and test data for this project are available on:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

Questions addressing

HOW WAS BUILT THE MODEL (Chosen Regressions method) ?

```
I chose the two most common method to make this analysis: The CART (Classification and regression T
- CARET method is easy to interpret, fast to run but hard to estimate the uncertainty
- RANDOM FOREST has a very good accuracy, but difficult to interpret and low speed  to run (may tak
```

HOW WAS CHOSEN THE CROSS VALIDATION?

```
There are several methods to estimate the model accuracy among them, Data Split, Boostrap, K-Fold C
- Data Splitting: Involves partitioning the data into an explicit Training dataset used to prepare 
- K-Fold Cross Validation; Involves splitting the dataset into K-subsets. For each subset is held o
- Leave One Out Cross Validation: a data instance is left out and a model constructed on all other 
```

WHAT YOU THINK THE EXPECTED OUT OF SAMPLE ERROR IS (= 1 - ACCURACY)?

```
    The two methods chosen provided diferente accuracy. the CART Method provided an accuracy of 60,65%,
```

USE THE FINAL PREDICT MODEL TO PREDICT 20 DIFFERENT TEST CASES.
See TABLE A, in the last session

Data Analysis

Getting and Cleaning Data

Loading the libraries necessary to run the codes

```r
library(dplyr)
library(caret)
library(rattle)
library(ggplot2)
library(rpart)
library(rpart.plot)
library(randomForest)
```

Loading the files necessary to develop the project

```r
training_original <- read.csv("J:/pml-training.csv", sep=";", stringsAsFactor = FALSE,na.strings=c("#DIV
testing_original  <- read.csv("J:/pml-testing.csv",  sep=";", stringsAsFactor = FALSE,na.strings=c("#DIV
```

```r
str(training_original)
```

```
## 'data.frame':    19622 obs. of  160 variables:
##  $ X                       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name               : chr  "carlitos" "carlitos" "carlitos" "carlitos" ...
##  $ raw_timestamp_part_1    : int  1323084231 1323084231 1323084231 1323084232 1323084232 1323084232 1
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 304277 368296 440390 484323 4844
##  $ cvtd_timestamp          : chr  "05/12/2011 11:23" "05/12/2011 11:23" "05/12/2011 11:23" "05/12/201
##  $ new_window              : chr  "no" "no" "no" "no" ...
##  $ num_window              : int  11 11 11 12 12 12 12 12 12 12 ...
##  $ roll_belt               : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.45 ...
##  $ pitch_belt              : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.17 ...
##  $ yaw_belt                : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 ...
##  $ total_accel_belt        : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ kurtosis_roll_belt      : chr  "" "" "" "" ...
##  $ kurtosis_picth_belt     : chr  "" "" "" "" ...
##  $ kurtosis_yaw_belt       : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_belt      : chr  "" "" "" "" ...
##  $ skewness_roll_belt.1    : chr  "" "" "" "" ...
##  $ skewness_yaw_belt       : logi  NA NA NA NA NA NA ...
##  $ max_roll_belt           : chr  "NA" "NA" "NA" "NA" ...
##  $ max_picth_belt          : chr  "NA" "NA" "NA" "NA" ...
##  $ max_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_roll_belt           : chr  "NA" "NA" "NA" "NA" ...
##  $ min_pitch_belt          : chr  "NA" "NA" "NA" "NA" ...
##  $ min_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_roll_belt     : chr  "NA" "NA" "NA" "NA" ...
##  $ amplitude_pitch_belt    : chr  "NA" "NA" "NA" "NA" ...
##  $ amplitude_yaw_belt      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_total_accel_belt    : chr  "NA" "NA" "NA" "NA" ...
##  $ avg_roll_belt           : chr  "NA" "NA" "NA" "NA" ...
##  $ stddev_roll_belt        : chr  "NA" "NA" "NA" "NA" ...
##  $ var_roll_belt           : chr  "NA" "NA" "NA" "NA" ...
##  $ avg_pitch_belt          : chr  "NA" "NA" "NA" "NA" ...
##  $ stddev_pitch_belt       : chr  "NA" "NA" "NA" "NA" ...
##  $ var_pitch_belt          : chr  "NA" "NA" "NA" "NA" ...
##  $ avg_yaw_belt            : chr  "NA" "NA" "NA" "NA" ...
##  $ stddev_yaw_belt         : chr  "NA" "NA" "NA" "NA" ...
##  $ var_yaw_belt            : chr  "NA" "NA" "NA" "NA" ...
##  $ gyros_belt_x            : num  0 0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 0.03 ...
##  $ gyros_belt_y            : num  0 0 0 0 0.02 0 0 0 0 0 ...
##  $ gyros_belt_z            : num  -0.02 -0.02 -0.02 -0.03 -0.02 -0.02 -0.02 -0.02 -0.02 0 ...
##  $ accel_belt_x            : int  -21 -22 -20 -22 -21 -21 -22 -22 -20 -21 ...
##  $ accel_belt_y            : int  4 4 5 3 2 4 3 4 2 4 ...
##  $ accel_belt_z            : int  22 22 23 21 24 21 21 21 24 22 ...
##  $ magnet_belt_x           : int  -3 -7 -2 -6 -6 0 -4 -2 1 -3 ...
##  $ magnet_belt_y           : int  599 608 600 604 600 603 599 603 602 609 ...
##  $ magnet_belt_z           : int  -313 -311 -305 -310 -302 -312 -311 -313 -312 -308 ...
##  $ roll_arm                : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -128 ...
##  $ pitch_arm               : num  22.5 22.5 22.5 22.1 22.1 22 21.9 21.8 21.7 21.6 ...
##  $ yaw_arm                 : num  -161 -161 -161 -161 -161 -161 -161 -161 -161 -161 ...
##  $ total_accel_arm         : int  34 34 34 34 34 34 34 34 34 34 ...
##  $ var_accel_arm           : chr  "NA" "NA" "NA" "NA" ...
##  $ avg_roll_arm            : chr  "NA" "NA" "NA" "NA" ...
##  $ stddev_roll_arm         : chr  "NA" "NA" "NA" "NA" ...
##  $ var_roll_arm            : chr  "NA" "NA" "NA" "NA" ...
##  $ avg_pitch_arm           : chr  "NA" "NA" "NA" "NA" ...
##  $ stddev_pitch_arm        : chr  "NA" "NA" "NA" "NA" ...
##  $ var_pitch_arm           : chr  "NA" "NA" "NA" "NA" ...
##  $ avg_yaw_arm             : chr  "NA" "NA" "NA" "NA" ...
##  $ stddev_yaw_arm          : chr  "NA" "NA" "NA" "NA" ...
##  $ var_yaw_arm             : chr  "NA" "NA" "NA" "NA" ...
##  $ gyros_arm_x             : num  0 0.02 0.02 0.02 0 0.02 0 0.02 0.02 0.02 ...
##  $ gyros_arm_y             : num  0 -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -0.02 -0.03 -0.03 ...
##  $ gyros_arm_z             : num  -0.02 -0.02 -0.02 0.02 0 0 0 0 -0.02 -0.02 ...
##  $ accel_arm_x             : int  -288 -290 -289 -289 -289 -289 -289 -289 -288 -288 ...
##  $ accel_arm_y             : int  109 110 110 111 111 111 111 111 109 110 ...
##  $ accel_arm_z             : int  -123 -125 -126 -123 -123 -122 -125 -124 -122 -124 ...
##  $ magnet_arm_x            : int  -368 -369 -368 -372 -374 -369 -373 -372 -369 -376 ...
##  $ magnet_arm_y            : int  337 337 344 344 337 342 336 338 341 334 ...
##  $ magnet_arm_z            : int  516 513 513 512 506 513 509 510 518 516 ...
##  $ kurtosis_roll_arm       : chr  "" "" "" "" ...
##  $ kurtosis_picth_arm      : chr  "" "" "" "" ...
##  $ kurtosis_yaw_arm        : chr  "" "" "" "" ...
##  $ 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            : chr  "NA" "NA" "NA" "NA" ...
##  $ max_picth_arm           : chr  "NA" "NA" "NA" "NA" ...
##  $ max_yaw_arm             : chr  "NA" "NA" "NA" "NA" ...
##  $ min_roll_arm            : chr  "NA" "NA" "NA" "NA" ...
##  $ min_pitch_arm          : chr  "NA" "NA" "NA" "NA" ...
##  $ min_yaw_arm            : chr  "NA" "NA" "NA" "NA" ...
##  $ amplitude_roll_arm     : chr  "NA" "NA" "NA" "NA" ...
##  $ amplitude_pitch_arm    : chr  "NA" "NA" "NA" "NA" ...
##  $ amplitude_yaw_arm      : chr  "NA" "NA" "NA" "NA" ...
##  $ roll_dumbbell          : chr  "1.305.217.456" "1.313.073.959" "1.285.074.981" "1.343.119.971" ..
##  $ pitch_dumbbell         : chr  "-7.049.400.371" "-7.063.750.507" "-7.027.811.982" "-7.039.379.464
##  $ yaw_dumbbell           : chr  "-8.487.393.888" "-8.471.064.711" "-8.514.078.134" "-8.487.362.553
##  $ 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      : chr  "NA" "NA" "NA" "NA" ...
##  $ max_picth_dumbbell     : chr  "NA" "NA" "NA" "NA" ...
##  $ max_yaw_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_roll_dumbbell      : chr  "NA" "NA" "NA" "NA" ...
##  $ min_pitch_dumbbell     : chr  "NA" "NA" "NA" "NA" ...
##  $ min_yaw_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_roll_dumbbell: chr  "NA" "NA" "NA" "NA" ...
##   [list output truncated]
```

The project has the following dimensions (number of rows versus number of columns)

```r
dim(training_original)
```

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

In order to manipulate the data to perform the regression and correlation analysis, the columns of the project should be converted to numeric, the conversion of the 160 columns was done using the "FOR" loop excluding only the column "classe".

```r
a<- ncol(training_original)
for (i in 1:(a-1)){training_original[,i] <- as.numeric(training_original[,i])}
b<- ncol(testing_original)
for (i in 1:(b-1)){testing_original [,i] <- as.numeric(testing_original [,i])}
```

The variables which the sum of the column is zero were excluded.

```r
training_original <- training_original[, colSums(is.na(training_original)) ==0]
```

The variables which the variation od the column is zero also were excluded.

```r
classe  <- training_original[,ncol(training_original)]
zeroVar <- nearZeroVar(training_original, saveMetrics=TRUE)
training_original <- training_original[, zeroVar$nzv==FALSE]
```

The variables with high correlation with another variable was removed using the collinearity approach.

```
#Removing columns with collinearity
correlation <- cor(training_original[,-ncol(training_original)])
top <- findCorrelation( correlation, cutoff =.75)
training_original <- training_original[, -top]
```

Some variables were eliminated based on their null influence on the final results.

```
training_original <- select(training_original, -1,-2,-3,-4)
training_original$classe <- as.factor(training_original$classe)
```

In order to maintain the project reproducible, it was utilized the set.seed (1000).

```
set.seed(1000)
```

The data was split 70% for the training dataset and 30% for the testing dataset.

```
inTrain  <-  createDataPartition(training_original$classe, p=0.7, list=FALSE)
training <-  training_original[inTrain,]
testing  <-  training_original[-inTrain,]
```

CART (Classification and regression Tree) RESULTS.

```
modFit_T<- train(classe~., method="rpart", data= training)
modFit_T
```

```
## CART
##
## 13737 samples
##    31 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
##
## Summary of sample sizes: 13737, 13737, 13737, 13737, 13737, 13737, ...
##
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa      Accuracy SD  Kappa SD
##   0.01759740  0.6081001  0.5033186  0.04442749   0.05680087
##   0.02120842  0.5720932  0.4582851  0.03226512   0.04151583
##   0.03092259  0.4261071  0.2257590  0.13692269   0.22390158
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.0175974.
```

```
print(modFit_T$finalModel)
```

```
## n= 13737
##
```

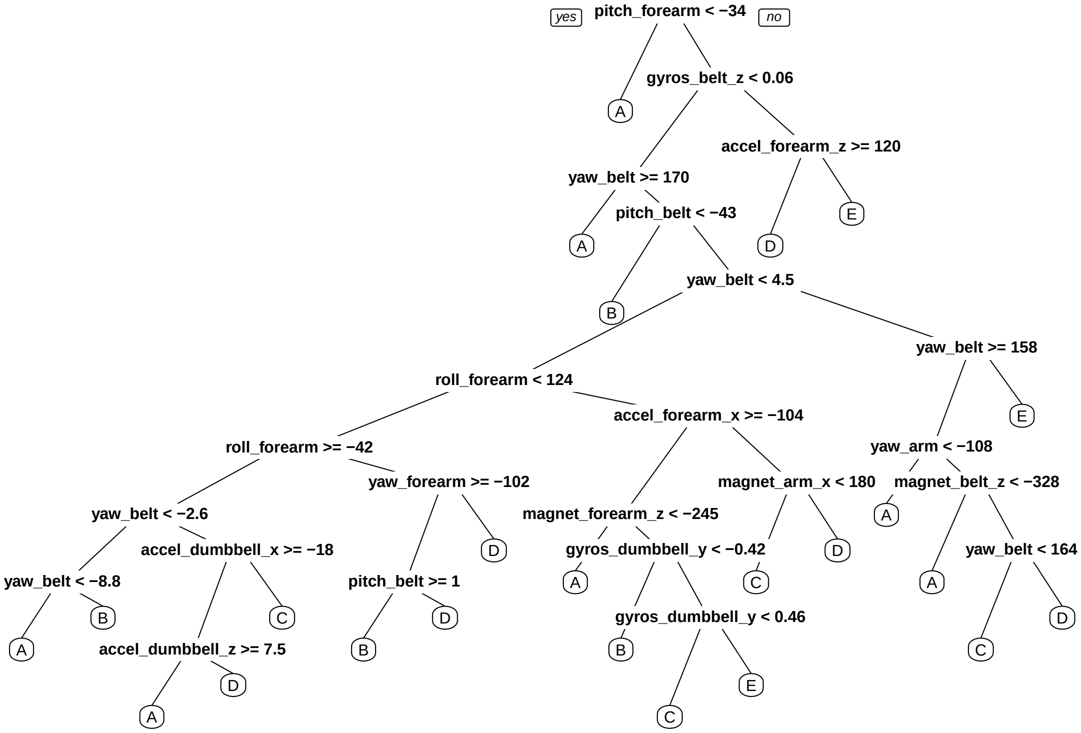
```
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##   1) root 13737 9831 A (0.28 0.19 0.17 0.16 0.18)
##     2) pitch_forearm< -33.95 1101    7 A (0.99 0.0064 0 0 0) *
##     3) pitch_forearm>=-33.95 12636 9824 A (0.22 0.21 0.19 0.18 0.2)
##       6) gyros_belt_z< 0.06 11792 8989 A (0.24 0.22 0.2 0.18 0.16)
##        12) yaw_belt>=169.5 580   58 A (0.9 0.04 0 0.052 0.0086) *
##        13) yaw_belt< 169.5 11212 8623 B (0.2 0.23 0.21 0.18 0.17)
##          26) pitch_belt< -42.95 563   78 B (0.016 0.86 0.089 0.016 0.018) *
##          27) pitch_belt>=-42.95 10649 8309 C (0.21 0.2 0.22 0.19 0.18)
##            54) yaw_belt< 4.515 8569 6507 B (0.23 0.24 0.22 0.2 0.11)
##             108) roll_forearm< 124.5 4763 3078 A (0.35 0.25 0.13 0.2 0.061)
##               216) roll_forearm>=-42.25 2580 1210 A (0.53 0.23 0.12 0.12 0.0027) *
##               217) roll_forearm< -42.25 2183 1529 D (0.14 0.28 0.15 0.3 0.13)
##                 434) yaw_forearm>=-101.5 1392  886 B (0.16 0.36 0.18 0.13 0.17) *
##                 435) yaw_forearm< -101.5 791  315 D (0.12 0.12 0.091 0.6 0.062) *
##             109) roll_forearm>=124.5 3806 2568 C (0.075 0.23 0.33 0.21 0.17)
##               218) accel_forearm_x>=-104.5 2652 1666 C (0.085 0.27 0.37 0.071 0.2)
##                 436) magnet_forearm_z< -245 217   43 A (0.8 0.18 0 0.014 0) *
##                 437) magnet_forearm_z>=-245 2435 1449 C (0.021 0.28 0.4 0.076 0.22)
##                   874) gyros_dumbbell_y< -0.425 486  185 B (0.035 0.62 0.12 0.037 0.19) *
##                   875) gyros_dumbbell_y>=-0.425 1949 1021 C (0.017 0.2 0.48 0.086 0.22) *
##               219) accel_forearm_x< -104.5 1154  562 D (0.053 0.11 0.22 0.51 0.1) *
##            55) yaw_belt>=4.515 2080 1120 E (0.14 0.02 0.23 0.15 0.46)
##             110) yaw_belt>=158.5 1122  652 C (0.27 0.037 0.42 0.27 0.0053) *
##             111) yaw_belt< 158.5 958    4 E (0 0 0 0.0042 1) *
##       7) gyros_belt_z>=0.06 844  227 E (0.011 0.046 0.0071 0.2 0.73) *
```

```
predictions_T <- predict(modFit_T, newdata=testing)
confusionMatrix(predictions_T, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1317  293  145  134    4
##          B  126  535  136   96  139
##          C  160  190  609  224  194
##          D   69  102  133  426   63
##          E    2   19    3   84  682
##
## Overall Statistics
##
##                Accuracy : 0.6065
##                  95% CI : (0.5938, 0.619)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7867  0.46971   0.5936  0.44191   0.6303
## Specificity            0.8632  0.89528   0.8419  0.92542   0.9775
## Pos Pred Value         0.6957  0.51841   0.4423  0.53720   0.8633
## Neg Pred Value         0.9106  0.87554   0.9075  0.89434   0.9215
## Prevalence             0.2845  0.19354   0.1743  0.16381   0.1839
## Detection Rate         0.2238  0.09091   0.1035  0.07239   0.1159
## Detection Prevalence   0.3217  0.17536   0.2340  0.13475   0.1342
## Balanced Accuracy      0.8250  0.68250   0.7178  0.68367   0.8039
```

```r
cart <- rpart(classe~., data=training, method="class")
prp(cart)
```



RANDOM FOREST RESULTS.

```r
modFit_RF<- train(classe~.,  method="rf", data = training)
modFit_RF
```

```
## Random Forest
##
## 13737 samples
##    31 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
##
## Summary of sample sizes: 13737, 13737, 13737, 13737, 13737, 13737, ...
##
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9879621  0.9847667  0.001334410  0.001689145
##   16    0.9858949  0.9821525  0.001979898  0.002501480
##   31    0.9761664  0.9698456  0.003464061  0.004372205
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

```
print(modFit_RF$finalModel)
```

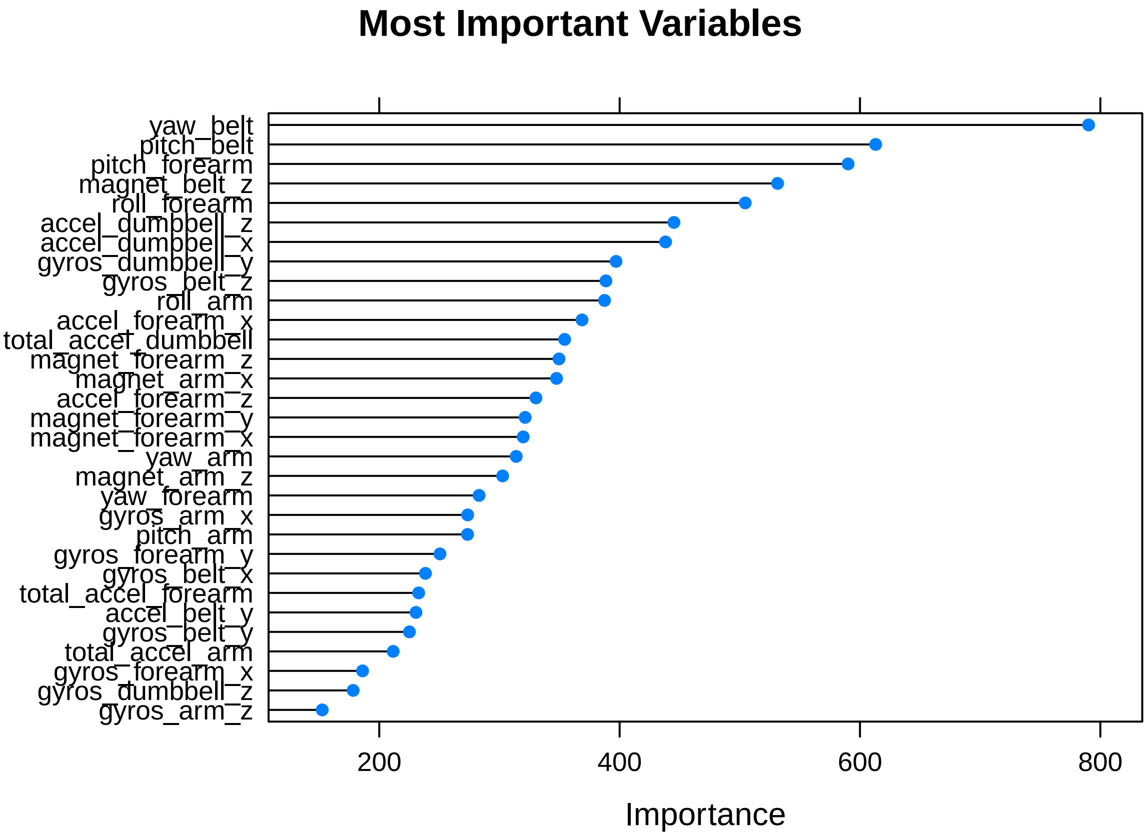
```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 0.71%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3899    3    1    1    2 0.001792115
## B    9 2645    3    0    1 0.004890895
## C    0   22 2361   13    0 0.014607679
## D    0    0   38 2212    2 0.017761989
## E    0    1    0    2 2522 0.001188119
```

```
predictions_RF <- predict(modFit_RF, newdata=testing)
confusionMatrix(predictions_RF, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1671    5    0    0    0
##          B    3 1132    8    0    0
##          C    0    1 1014   15    0
##          D    0    0    4  949    3
##          E    0    1    0    0 1079
##
## Overall Statistics
##
##                Accuracy : 0.9932
##                  95% CI : (0.9908, 0.9951)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9914
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9939   0.9883   0.9844   0.9972
## Specificity            0.9988   0.9977   0.9967   0.9986   0.9998
## Pos Pred Value         0.9970   0.9904   0.9845   0.9927   0.9991
## Neg Pred Value         0.9993   0.9985   0.9975   0.9970   0.9994
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2839   0.1924   0.1723   0.1613   0.1833
## Detection Prevalence   0.2848   0.1942   0.1750   0.1624   0.1835
## Balanced Accuracy      0.9985   0.9958   0.9925   0.9915   0.9985
```

Most Important Variables

```
varimp <- varImp(modFit_RF, scale=FALSE)
plot(varimp, main="Most Important Variables")
```



```
head(getTree(modFit_RF$finalModel, k=2))
```

```
##   left daughter right daughter split var split point status prediction
## 1             2              3        29     -18.500      1          0
## 2             4              5         1     -42.550      1          0
## 3             6              7        11      31.500      1          0
## 4              8              9         6      42.500      1           0
## 5             10             11         4       0.135      1           0
## 6             12             13         7    -497.500      1           0
```

```
pred<-testing$classe
table(pred, testing$classe)
```

```
##
## pred    A    B    C    D    E
##    A 1674    0    0    0    0
##    B    0 1139    0    0    0
##    C    0    0 1026    0    0
##    D    0    0    0  964    0
##    E    0    0    0    0 1082
```

The final predict model with better accuracy (Randon Forest) was used to predict the 20 different test cases proposed in the initial request of the project

```
predictions_RR <- predict(modFit_RF, newdata=testing_original)
```

Table A: showing the prediction of the 20 different test cases

```
table(predictions_RR)
```

```
## predictions_RR
## A B C D E
## 7 8 1 1 3
```

```
predictions_RR
```

```
##  [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
```

sources:

http://groupware.les.inf.puc-rio.br/har. https://citizennet.com/blog/2012/11/10/random-forests-ensembles-and-performance-
http://machinelearningmastery.com/how-to-estimate-model-accuracy-in-r-using-the-caret-package/