Machine Learning Project 1

Phillip

September 26, 2015

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. In this project, your goal will be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. More information is available from the website here: http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har) (see the section on the Weight Lifting Exercise Dataset).

Objective

The goal of the project is to predict the manner in which they did the exercise. This is the “classe” variable in the training set.Use any of the other variables to predict with. Create a report describing how the model is built , how cross validation is used, and what was the expected out of sample error is, and choices were made. And using the prediction model to predict 20 different test cases.

Data

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

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

(Class A) exactly according the the specification (Class B) throwing the elbows to the front (Class C) lifting the dumbbell only halfway (Class D) lowering the dumbbell only halfway (Class E) throwing the hips to the front

Data Source (Citation)

Ugulino, W.; Cardador, D.; Vega, K.; Velloso, E.; Milidiu, R.; Fuks, H. Wearable Computing: Accelerometers’ Data Classification of Body Postures and Movements. Proceedings of 21st Brazilian Symposium on Artificial Intelligence. Advances in Artificial Intelligence - SBIA 2012. In: Lecture Notes in

Computer Science. , pp. 52-61. Curitiba, PR: Springer Berlin / Heidelberg, 2012. ISBN 978-3-642-34458-9. DOI: 10.1007/978-3-642-34459-6_6. Cited by 2 (Google Scholar)

Read more: http://groupware.les.inf.puc-rio.br/har#ixzz3mu8MFhoA (http://groupware.les.inf.puc-rio.br/har#ixzz3mu8MFhoA)

```
## Loading required package: lattice
## Loading required package: ggplot2
## Loading required package: dplyr
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
## Loading required package: RGtk2
## Rattle: A free graphical interface for data mining with R.
## Version 3.5.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
## R version 3.2.2 (2015-08-14)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 7 x64 (build 7601) Service Pack 1
##
## locale:
## [1] LC_COLLATE=English_United States.1252
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] rattle_3.5.0        RGtk2_2.20.31       corrplot_0.73
##  [4] randomForest_4.6-10 rpart.plot_1.5.2    rpart_4.1-10
##  [7] kernlab_0.9-22      caret_6.0-52        ggplot2_1.0.1
## [10] lattice_0.20-33
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.0         formatR_1.2         nloptr_1.0.4
##  [4] plyr_1.8.3          iterators_1.0.7     tools_3.2.2
##  [7] digest_0.6.8        lme4_1.1-9          evaluate_0.7.2
## [10] nlme_3.1-122        gtable_0.1.2        mgcv_1.8-7
## [13] Matrix_1.2-2        foreach_1.4.2       yaml_2.1.13
## [16] parallel_3.2.2      brglm_0.5-9         SparseM_1.7
## [19] proto_0.3-10        BradleyTerry2_1.0-6 stringr_1.0.0
## [22] knitr_1.11          MatrixModels_0.4-1  gtools_3.5.0
## [25] stats4_3.2.2        grid_3.2.2          nnet_7.3-11
## [28] rmarkdown_0.8       minqa_1.2.4         reshape2_1.4.1
## [31] car_2.1-0           magrittr_1.5        scales_0.3.0
## [34] codetools_0.2-14    htmltools_0.2.6     MASS_7.3-44
## [37] splines_3.2.2       pbkrtest_0.4-2      colorspace_1.2-6
## [40] quantreg_5.19       stringi_0.5-5       munsell_0.4.2
```

Data Exploratory

Checking for the dimensions of the Training and Testing data set.

```
# Check data size(rows)

dim(training)
```

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

```
dim(testing)
```

```
## [1]  20 160
```

```
str(training)
```

```
## 'data.frame':    19622 obs. of  160 variables:
##  $ X                       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name               : Factor w/ 6 levels "adelmo","carlitos",..: 2 2 2 2 2 2 2
2 2 2 ...
##  $ raw_timestamp_part_1    : int  1323084231 1323084231 1323084231 1323084232 13230842
32 1323084232 1323084232 1323084232 1323084232 1323084232 ...
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 304277 368296 440
390 484323 484434 ...
##  $ cvtd_timestamp          : Factor w/ 20 levels "02/12/2011 13:32",..: 9 9 9 9 9 9 9
9 9 9 ...
##  $ new_window              : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ num_window              : int  11 11 11 12 12 12 12 12 12 12 ...
##  $ roll_belt               : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.45
...
##  $ pitch_belt              : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.17
...
##  $ yaw_belt                : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -9
4.4 -94.4 ...
##  $ total_accel_belt        : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ kurtosis_roll_belt      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_picth_belt     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_yaw_belt       : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_belt      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_roll_belt.1    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_yaw_belt       : logi  NA NA NA NA NA NA ...
##  $ max_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_belt          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_belt          : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_roll_belt     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_belt    : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_belt      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_total_accel_belt    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_belt        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_belt           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_belt       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_belt          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_belt         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_belt            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ gyros_belt_x            : num  0 0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 0.03 ...
##  $ gyros_belt_y            : num  0 0 0 0 0.02 0 0 0 0 0 ...
##  $ gyros_belt_z            : num  -0.02 -0.02 -0.02 -0.03 -0.02 -0.02 -0.02 -0.02 -0.0
2 0 ...
##  $ accel_belt_x            : int  -21 -22 -20 -22 -21 -21 -22 -22 -20 -21 ...
##  $ accel_belt_y            : int  4 4 5 3 2 4 3 4 2 4 ...
```

```
##  $ accel_belt_z           : int  22 22 23 21 24 21 21 21 24 22 ...
##  $ magnet_belt_x          : int  -3 -7 -2 -6 -6 0 -4 -2 1 -3 ...
##  $ magnet_belt_y          : int  599 608 600 604 600 603 599 603 602 609 ...
##  $ magnet_belt_z          : int  -313 -311 -305 -310 -302 -312 -311 -313 -312 -308
...
##  $ roll_arm               : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -128
...
##  $ pitch_arm              : num  22.5 22.5 22.5 22.1 22.1 22 21.9 21.8 21.7 21.6 ...
##  $ yaw_arm                : num  -161 -161 -161 -161 -161 -161 -161 -161 -161 -161
...
##  $ total_accel_arm        : int  34 34 34 34 34 34 34 34 34 34 ...
##  $ var_accel_arm          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_arm        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_arm          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_arm       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_arm          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_arm         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_arm            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ gyros_arm_x            : num  0 0.02 0.02 0.02 0 0.02 0 0.02 0.02 0.02 ...
##  $ gyros_arm_y            : num  0 -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -0.02 -0.03
-0.03 ...
##  $ gyros_arm_z            : num  -0.02 -0.02 -0.02 0.02 0 0 0 0 -0.02 -0.02 ...
##  $ accel_arm_x            : int  -288 -290 -289 -289 -289 -289 -289 -289 -288 -288
...
##  $ accel_arm_y            : int  109 110 110 111 111 111 111 111 109 110 ...
##  $ accel_arm_z            : int  -123 -125 -126 -123 -123 -122 -125 -124 -122 -124
...
##  $ magnet_arm_x           : int  -368 -369 -368 -372 -374 -369 -373 -372 -369 -376
...
##  $ magnet_arm_y           : int  337 337 344 344 337 342 336 338 341 334 ...
##  $ magnet_arm_z           : int  516 513 513 512 506 513 509 510 518 516 ...
##  $ kurtosis_roll_arm      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_picth_arm     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_yaw_arm       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_roll_arm      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_pitch_arm     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_yaw_arm       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_roll_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_arm          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_arm            : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_roll_arm           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_arm          : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_arm            : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_roll_arm     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_arm    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_arm      : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ roll_dumbbell          : num  13.1 13.1 12.9 13.4 13.4 ...
```

```
##  $ pitch_dumbbell          : num  -70.5 -70.6 -70.3 -70.4 -70.4 ...
##  $ yaw_dumbbell            : num  -84.9 -84.7 -85.1 -84.9 -84.9 ...
##  $ kurtosis_roll_dumbbell  : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_picth_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ kurtosis_yaw_dumbbell   : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_dumbbell  : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_pitch_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ skewness_yaw_dumbbell   : logi  NA NA NA NA NA NA ...
##  $ max_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_dumbbell        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_dumbbell        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_roll_dumbbell : num  NA NA NA NA NA NA NA NA NA NA ...
##   [list output truncated]
```

From the str() function, it is noticed that many columns/variables are contains “NA”.

Data PreProcessing

A check on the training and testing data (trainCl & testCl), shows that the columns have been trimmed from 160 to 60.

```
## Remove columns that constins many NA's values
trainCl <- training[, colSums(is.na(training)) == 0]
testCl <- testing[, colSums(is.na(testing)) == 0]

dim(trainCl)
```

```
## [1] 19622    60
```

```
dim(testCl)
```

```
## [1] 20 60
```

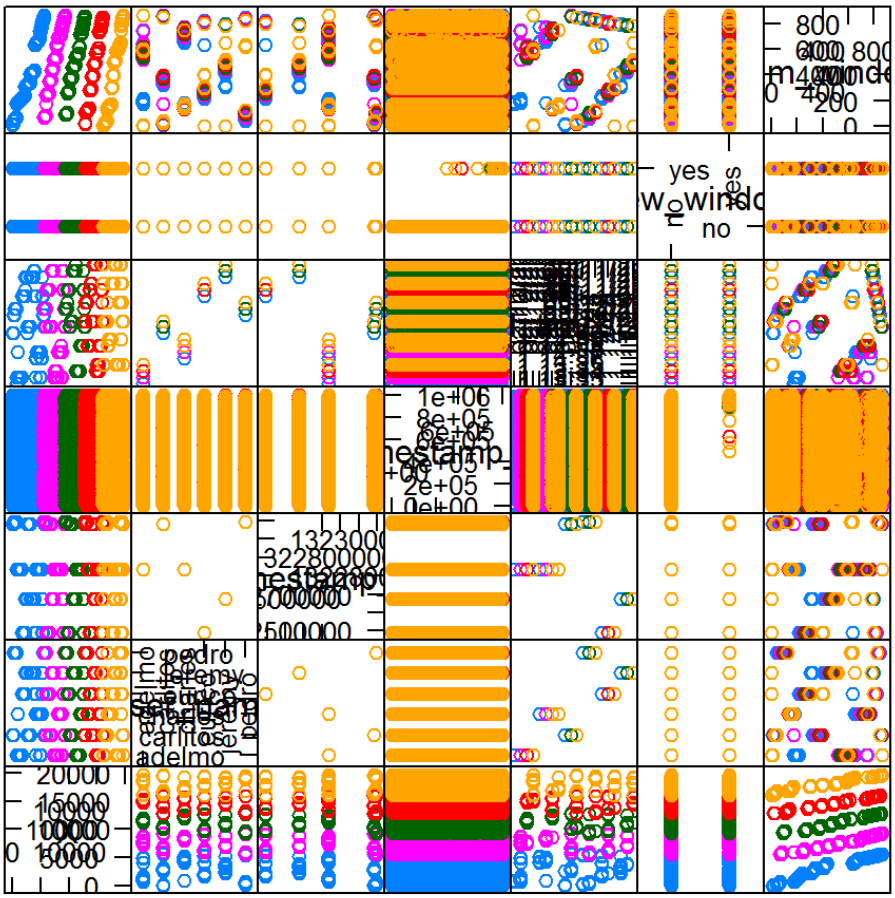
Next, using the str() function to check on the remaining data training & test data shows that the 1st 7 columns of data contains data may not be useful predictor of Classe.

```
str(trainCl)
```

```
## 'data.frame':    19622 obs. of  60 variables:
##  $ X                    : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name            : Factor w/ 6 levels "adelmo","carlitos",..: 2 2 2 2 2 2 2 2 2
2 ...
##  $ raw_timestamp_part_1: int  1323084231 1323084231 1323084231 1323084232 1323084232 1
323084232 1323084232 1323084232 1323084232 1323084232 ...
##  $ raw_timestamp_part_2: int  788290 808298 820366 120339 196328 304277 368296 440390
484323 484434 ...
##  $ cvtd_timestamp       : Factor w/ 20 levels "02/12/2011 13:32",..: 9 9 9 9 9 9 9 9 9
9 ...
##  $ new_window           : Factor w/ 2 levels "no","yes": 1 1 1 1 1 1 1 1 1 1 ...
##  $ num_window           : int  11 11 11 12 12 12 12 12 12 12 ...
##  $ roll_belt            : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.45 ...
##  $ pitch_belt           : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.17 ...
##  $ yaw_belt             : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -9
4.4 ...
##  $ total_accel_belt     : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ gyros_belt_x         : num  0 0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 0.03 ...
##  $ gyros_belt_y         : num  0 0 0 0 0.02 0 0 0 0 0 ...
##  $ gyros_belt_z         : num  -0.02 -0.02 -0.02 -0.03 -0.02 -0.02 -0.02 -0.02 -0.02 0
...
##  $ accel_belt_x         : int  -21 -22 -20 -22 -21 -21 -22 -22 -20 -21 ...
##  $ accel_belt_y         : int  4 4 5 3 2 4 3 4 2 4 ...
##  $ accel_belt_z         : int  22 22 23 21 24 21 21 21 24 22 ...
##  $ magnet_belt_x        : int  -3 -7 -2 -6 -6 0 -4 -2 1 -3 ...
##  $ magnet_belt_y        : int  599 608 600 604 600 603 599 603 602 609 ...
##  $ magnet_belt_z        : int  -313 -311 -305 -310 -302 -312 -311 -313 -312 -308 ...
##  $ roll_arm             : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -128 ...
##  $ pitch_arm            : num  22.5 22.5 22.5 22.1 22.1 22 21.9 21.8 21.7 21.6 ...
##  $ yaw_arm              : num  -161 -161 -161 -161 -161 -161 -161 -161 -161 -161 ...
##  $ total_accel_arm      : int  34 34 34 34 34 34 34 34 34 34 ...
##  $ gyros_arm_x          : num  0 0.02 0.02 0.02 0 0.02 0 0.02 0.02 0.02 ...
##  $ gyros_arm_y          : num  0 -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -0.02 -0.03 -0.03
...
##  $ gyros_arm_z          : num  -0.02 -0.02 -0.02 0.02 0 0 0 0 -0.02 -0.02 ...
##  $ accel_arm_x          : int  -288 -290 -289 -289 -289 -289 -289 -289 -288 -288 ...
##  $ accel_arm_y          : int  109 110 110 111 111 111 111 111 109 110 ...
##  $ accel_arm_z          : int  -123 -125 -126 -123 -123 -122 -125 -124 -122 -124 ...
##  $ magnet_arm_x         : int  -368 -369 -368 -372 -374 -369 -373 -372 -369 -376 ...
##  $ magnet_arm_y         : int  337 337 344 344 337 342 336 338 341 334 ...
##  $ magnet_arm_z         : int  516 513 513 512 506 513 509 510 518 516 ...
##  $ roll_dumbbell        : num  13.1 13.1 12.9 13.4 13.4 ...
##  $ pitch_dumbbell       : num  -70.5 -70.6 -70.3 -70.4 -70.4 ...
##  $ yaw_dumbbell         : num  -84.9 -84.7 -85.1 -84.9 -84.9 ...
##  $ total_accel_dumbbell: int  37 37 37 37 37 37 37 37 37 37 ...
##  $ gyros_dumbbell_x     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ gyros_dumbbell_y     : num  -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02
-0.02 ...
##  $ gyros_dumbbell_z     : num  0 0 0 -0.02 0 0 0 0 0 0 ...
##  $ accel_dumbbell_x     : int  -234 -233 -232 -232 -233 -234 -232 -234 -232 -235 ...
```

```
##  $ accel_dumbbell_y    : int  47 47 46 48 48 48 47 46 47 48 ...
##  $ accel_dumbbell_z    : int  -271 -269 -270 -269 -270 -269 -270 -272 -269 -270 ...
##  $ magnet_dumbbell_x   : int  -559 -555 -561 -552 -554 -558 -551 -555 -549 -558 ...
##  $ magnet_dumbbell_y   : int  293 296 298 303 292 294 295 300 292 291 ...
##  $ magnet_dumbbell_z   : num  -65 -64 -63 -60 -68 -66 -70 -74 -65 -69 ...
##  $ roll_forearm        : num  28.4 28.3 28.3 28.1 28 27.9 27.9 27.8 27.7 27.7 ...
##  $ pitch_forearm       : num  -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.8 -63.8 -6
3.8 ...
##  $ yaw_forearm         : num  -153 -153 -152 -152 -152 -152 -152 -152 -152 -152 ...
##  $ total_accel_forearm : int  36 36 36 36 36 36 36 36 36 36 ...
##  $ gyros_forearm_x     : num  0.03 0.02 0.03 0.02 0.02 0.02 0.02 0.02 0.03 0.02 ...
##  $ gyros_forearm_y     : num  0 0 -0.02 -0.02 0 -0.02 0 -0.02 0 0 ...
##  $ gyros_forearm_z     : num  -0.02 -0.02 0 0 -0.02 -0.03 -0.02 0 -0.02 -0.02 ...
##  $ accel_forearm_x     : int  192 192 196 189 189 193 195 193 193 190 ...
##  $ accel_forearm_y     : int  203 203 204 206 206 203 205 205 204 205 ...
##  $ accel_forearm_z     : int  -215 -216 -213 -214 -214 -215 -215 -213 -214 -215 ...
##  $ magnet_forearm_x    : int  -17 -18 -18 -16 -17 -9 -18 -9 -16 -22 ...
##  $ magnet_forearm_y    : num  654 661 658 658 655 660 659 660 653 656 ...
##  $ magnet_forearm_z    : num  476 473 469 469 473 478 470 474 476 473 ...
##  $ classe              : Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1
...
```

```
featurePlot(x=trainCl[,c(1,2,3,4,5,6,7)],
         y = trainCl$classe,
         plot="pairs")
```



Scatter Plot Matrix

The above featureplot displays the correlation with the classe variable.

The 1st 7 columns are then removed.

```
# remove st 7 columns
trainCl <- subset(trainCl,select = -c(1:7))
testCl <- subset(testCl,select = -c(1:7))


dim(testCl)
```

```
## [1] 20 53
```

Data is partitioned

Here, the training data is partitioned into 75% training and 25% testing.

```
set.seed(123)
inTrain <- createDataPartition(trainCl$classe,p=.75,list=FALSE)
trainingCl <- trainCl[inTrain,]
testingCl <- trainCl[-inTrain,]

dim(trainingCl)
```

```
## [1] 14718    53
```

```
dim(testingCl)
```

```
## [1] 4904   53
```

Training the Model

Here, the model is trained using the Random Forest algorithm as it is rated as highly accurate. It was initialy trained using the caret library but it took very long and did not managed to complete. As such, the randomForest library is used - which is fast and managed to complete the task.

```
rfModel <- randomForest(classe ~.,data=trainingCl)
rfPredict <- predict(rfModel,newdata=testingCl)
confusionMatrix(rfPredict,testingCl$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1394    1    0    0    0
##          B    1  946    8    0    0
##          C    0    2  847    9    0
##          D    0    0    0  793    1
##          E    0    0    0    2  900
##
## Overall Statistics
##
##                Accuracy : 0.9951
##                  95% CI : (0.9927, 0.9969)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9938
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9993   0.9968   0.9906   0.9863   0.9989
## Specificity            0.9997   0.9977   0.9973   0.9998   0.9995
## Pos Pred Value         0.9993   0.9906   0.9872   0.9987   0.9978
## Neg Pred Value         0.9997   0.9992   0.9980   0.9973   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2843   0.1929   0.1727   0.1617   0.1835
## Detection Prevalence   0.2845   0.1947   0.1750   0.1619   0.1839
## Balanced Accuracy      0.9995   0.9973   0.9940   0.9930   0.9992
```

Sample Error Analysis

From the above statistics, the Accuracy : 0.9951 or 99.51% As such, the error is 1-0.9951 = 0.0049 or 0.49%

Test prediction model

```
#str(testCl)
result <- predict(rfModel, testCl[, -length(names(testCl))])
result
```

```
##  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
```

Save output to files

As required in the assignment

```
answers <- as.vector(result)

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