Practical Machine Learning Course Report

Kadri Umay

February 9, 2016

Introduction

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 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. 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).

The approach proposed for the Weight Lifting Exercises dataset is to investigate "how (well)" an activity was performed by the wearer. The "how (well)" investigation provides useful information for a large variety of applications,such as sports training.

The quality of execution is defined for Unilateral Dumbbell Biceps Curl in five different fashions. Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes. * A - Exactly according to the specification * 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

We will try to classify the quality of exeuction in one of the 5 different classes based on the data collected from the on-body sensors while the exercises were performed by six male participants aged between 20-28 years, with little weight lifting experience and simulating the mistakes.

Read more: http://groupware.les.inf.puc-rio.br/har#ixzz3z1Cxz0oe (http://groupware.les.inf.puc-rio.br/har#ixzz3z1Cxz0oe)

Data Preprocessing

```r
library(caret)
```

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

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

```
library(rpart)
library(rpart.plot)
library(randomForest)
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

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

```
library(corrplot)
library(nnet)
library(e1071)
library(MASS)
```

Set Directories and Download Data

Download Data and store in data frames

```
ProjectDir <- "c:\\users\\kadriu\\documents\\GitHub\\Coursera-Practical-Machine
-Learning"
SubDir <- "Data"
setwd(ProjectDir)

if (!file.exists(SubDir)) {
  dir.create(file.path(SubDir))
}
setwd(file.path(ProjectDir, SubDir))

TrainLink <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.c
sv"
TestLink <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.cs
v"
TrainFile <- "pml-training.csv"
TestFile <- "pml-testing.csv"
if (!file.exists(TrainFile))
  download.file(TrainLink, destfile = TrainFile, method = "curl")
if (!file.exists(TestFile))
  download.file(TestLink, destfile = TestFile, method = "curl")

RawTrainData <- read.csv(TrainFile)
RawTestData <- read.csv(TestFile)
```

Raw Training Data Summary The ‘classe’ variable in the dataset is the variable to predict

Number of Variables

```
dim(RawTrainData) #19622 160
```

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

```
str(RawTrainData, list.len=ncol(RawTrainData))
```

```
## '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 132308423
2 1323084232 1323084232 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 -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 0.02 0.02 0.0
3 ...
##  $ 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 -2
1 ...
##  $ 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 60
9 ...
##  $ magnet_belt_z          : int  -313 -311 -305 -310 -302 -312 -311 -313 -3
12 -308 ...
##  $ roll_arm               : num  -128 -128 -128 -128 -128 -128 -128 -128 -1
28 -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 -1
61 -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 0.0
2 ...
##  $ gyros_arm_y            : num  0 -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -0.0
2 -0.03 -0.03 ...
##  $ gyros_arm_z            : num  -0.02 -0.02 -0.02 0.02 0 0 0 0 -0.02 -0.0
2 ...
##  $ accel_arm_x            : int  -288 -290 -289 -289 -289 -289 -289 -289 -2
88 -288 ...
##  $ accel_arm_y            : int  109 110 110 111 111 111 111 111 109 11
0 ...
##  $ accel_arm_z            : int  -123 -125 -126 -123 -123 -122 -125 -124 -1
22 -124 ...
```

```
##  $ magnet_arm_x            : int  -368 -369 -368 -372 -374 -369 -373 -372 -3
69 -376 ...
##  $ magnet_arm_y            : int  337 337 344 344 337 342 336 338 341 33
4 ...
##  $ magnet_arm_z            : int  516 513 513 512 506 513 509 510 518 51
6 ...
##  $ 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","-0.007
3",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_picth_dumbbell : Factor w/ 401 levels "","-0.0163","-0.023
3",..: 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","-0.009
6",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_pitch_dumbbell : Factor w/ 402 levels "","-0.0053","-0.008
4",..: 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 ...
##  $ amplitude_pitch_dumbbell: num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_dumbbell  : Factor w/ 3 levels "","#DIV/0!","0.00": 1 1 1
1 1 1 1 1 1 1 ...
##  $ total_accel_dumbbell    : int  37 37 37 37 37 37 37 37 37 37 ...
##  $ var_accel_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_dumbbell    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_dumbbell       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_dumbbell   : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_dumbbell      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_dumbbell        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_dumbbell     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_dumbbell        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ 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 -2
32 -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 -2
69 -270 ...
##  $ magnet_dumbbell_x       : int  -559 -555 -561 -552 -554 -558 -551 -555 -5
49 -558 ...
##  $ magnet_dumbbell_y       : int  293 296 298 303 292 294 295 300 292 29
1 ...
##  $ magnet_dumbbell_z       : num  -65 -64 -63 -60 -68 -66 -70 -74 -65 -6
9 ...
##  $ 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 -63.8 ...
##  $ yaw_forearm             : num  -153 -153 -152 -152 -152 -152 -152 -152 -1
52 -152 ...
##  $ kurtosis_roll_forearm   : Factor w/ 322 levels "","-0.0227","-0.035
9",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_picth_forearm  : Factor w/ 323 levels "","-0.0073","-0.044
2",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ kurtosis_yaw_forearm    : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1
1 1 1 ...
##  $ skewness_roll_forearm   : Factor w/ 323 levels "","-0.0004","-0.001
3",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_pitch_forearm  : Factor w/ 319 levels "","-0.0113","-0.013
1",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ skewness_yaw_forearm    : Factor w/ 2 levels "","#DIV/0!": 1 1 1 1 1 1 1
1 1 1 ...
```

```
##  $ max_roll_forearm        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_picth_forearm       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ max_yaw_forearm         : Factor w/ 45 levels "","-0.1","-0.2",..: 1 1 1
1 1 1 1 1 1 1 ...
##  $ min_roll_forearm        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_pitch_forearm       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ min_yaw_forearm         : Factor w/ 45 levels "","-0.1","-0.2",..: 1 1 1
1 1 1 1 1 1 1 ...
##  $ amplitude_roll_forearm  : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_pitch_forearm : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ amplitude_yaw_forearm   : Factor w/ 3 levels "","#DIV/0!","0.00": 1 1 1
1 1 1 1 1 1 1 ...
##  $ total_accel_forearm     : int  36 36 36 36 36 36 36 36 36 36 ...
##  $ var_accel_forearm       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_roll_forearm        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_roll_forearm     : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_roll_forearm        : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_pitch_forearm       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_pitch_forearm    : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_pitch_forearm       : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ avg_yaw_forearm         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ stddev_yaw_forearm      : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ var_yaw_forearm         : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ 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 19
0 ...
##  $ accel_forearm_y         : int  203 203 204 206 206 203 205 205 204 20
5 ...
##  $ accel_forearm_z         : int  -215 -216 -213 -214 -214 -215 -215 -213 -2
14 -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 65
6 ...
##  $ magnet_forearm_z        : num  476 473 469 469 473 478 470 474 476 47
3 ...
##  $ classe                  : Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1
1 1 1 1 1 1 ...
```

Raw Test Data Summary Number of Rows

```
dim(RawTestData) #20 160
```

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

```
str(RawTestData, list.len=ncol(RawTestData))
```

```
## 'data.frame':    20 obs. of  160 variables:
##  $ X                       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name               : Factor w/ 6 levels "adelmo","carlitos",..: 6 5
5 1 4 5 5 5 2 3 ...
##  $ raw_timestamp_part_1    : int  1323095002 1322673067 1322673075 132283278
9 1322489635 1322673149 1322673128 1322673076 1323084240 1322837822 ...
##  $ raw_timestamp_part_2    : int  868349 778725 342967 560311 814776 510661
766645 54671 916313 384285 ...
##  $ cvtd_timestamp          : Factor w/ 11 levels "02/12/2011 13:33",..: 5 1
0 10 1 6 11 11 10 3 2 ...
##  $ new_window              : Factor w/ 1 level "no": 1 1 1 1 1 1 1 1 1 1 ...
##  $ num_window              : int  74 431 439 194 235 504 485 440 323 664 ...
##  $ roll_belt               : num  123 1.02 0.87 125 1.35 -5.92 1.2 0.43 0.9
3 114 ...
##  $ pitch_belt              : num  27 4.87 1.82 -41.6 3.33 1.59 4.44 4.15 6.7
2 22.4 ...
##  $ yaw_belt                : num  -4.75 -88.9 -88.5 162 -88.6 -87.7 -87.3 -8
8.5 -93.7 -13.1 ...
##  $ total_accel_belt        : int  20 4 5 17 3 4 4 4 4 18 ...
##  $ kurtosis_roll_belt      : logi  NA NA NA NA NA NA ...
##  $ kurtosis_picth_belt     : logi  NA NA NA NA NA NA ...
##  $ kurtosis_yaw_belt       : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_belt      : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_belt.1    : logi  NA NA NA NA NA NA ...
##  $ skewness_yaw_belt       : logi  NA NA NA NA NA NA ...
##  $ max_roll_belt           : logi  NA NA NA NA NA NA ...
##  $ max_picth_belt          : logi  NA NA NA NA NA NA ...
##  $ max_yaw_belt            : logi  NA NA NA NA NA NA ...
##  $ min_roll_belt           : logi  NA NA NA NA NA NA ...
##  $ min_pitch_belt          : logi  NA NA NA NA NA NA ...
##  $ min_yaw_belt            : logi  NA NA NA NA NA NA ...
##  $ amplitude_roll_belt     : logi  NA NA NA NA NA NA ...
##  $ amplitude_pitch_belt    : logi  NA NA NA NA NA NA ...
##  $ amplitude_yaw_belt      : logi  NA NA NA NA NA NA ...
##  $ var_total_accel_belt    : logi  NA NA NA NA NA NA ...
##  $ avg_roll_belt           : logi  NA NA NA NA NA NA ...
##  $ stddev_roll_belt        : logi  NA NA NA NA NA NA ...
##  $ var_roll_belt           : logi  NA NA NA NA NA NA ...
##  $ avg_pitch_belt          : logi  NA NA NA NA NA NA ...
##  $ stddev_pitch_belt       : logi  NA NA NA NA NA NA ...
##  $ var_pitch_belt          : logi  NA NA NA NA NA NA ...
##  $ avg_yaw_belt            : logi  NA NA NA NA NA NA ...
##  $ stddev_yaw_belt         : logi  NA NA NA NA NA NA ...
##  $ var_yaw_belt            : logi  NA NA NA NA NA NA ...
##  $ gyros_belt_x            : num  -0.5 -0.06 0.05 0.11 0.03 0.1 -0.06 -0.18
0.1 0.14 ...
##  $ gyros_belt_y            : num  -0.02 -0.02 0.02 0.11 0.02 0.05 0 -0.02 0
0.11 ...
```

```
##  $ gyros_belt_z          : num  -0.46 -0.07 0.03 -0.16 0 -0.13 0 -0.03 -0.
02 -0.16 ...
##  $ accel_belt_x          : int  -38 -13 1 46 -8 -11 -14 -10 -15 -25 ...
##  $ accel_belt_y          : int  69 11 -1 45 4 -16 2 -2 1 63 ...
##  $ accel_belt_z          : int  -179 39 49 -156 27 38 35 42 32 -158 ...
##  $ magnet_belt_x         : int  -13 43 29 169 33 31 50 39 -6 10 ...
##  $ magnet_belt_y         : int  581 636 631 608 566 638 622 635 600 60
1 ...
##  $ magnet_belt_z         : int  -382 -309 -312 -304 -418 -291 -315 -305 -3
02 -330 ...
##  $ roll_arm              : num  40.7 0 0 -109 76.1 0 0 0 -137 -82.4 ...
##  $ pitch_arm             : num  -27.8 0 0 55 2.76 0 0 0 11.2 -63.8 ...
##  $ yaw_arm               : num  178 0 0 -142 102 0 0 0 -167 -75.3 ...
##  $ total_accel_arm       : int  10 38 44 25 29 14 15 22 34 32 ...
##  $ var_accel_arm         : logi  NA NA NA NA NA NA ...
##  $ avg_roll_arm          : logi  NA NA NA NA NA NA ...
##  $ stddev_roll_arm       : logi  NA NA NA NA NA NA ...
##  $ var_roll_arm          : logi  NA NA NA NA NA NA ...
##  $ avg_pitch_arm         : logi  NA NA NA NA NA NA ...
##  $ stddev_pitch_arm      : logi  NA NA NA NA NA NA ...
##  $ var_pitch_arm         : logi  NA NA NA NA NA NA ...
##  $ avg_yaw_arm           : logi  NA NA NA NA NA NA ...
##  $ stddev_yaw_arm        : logi  NA NA NA NA NA NA ...
##  $ var_yaw_arm           : logi  NA NA NA NA NA NA ...
##  $ gyros_arm_x           : num  -1.65 -1.17 2.1 0.22 -1.96 0.02 2.36 -3.7
1 0.03 0.26 ...
##  $ gyros_arm_y           : num  0.48 0.85 -1.36 -0.51 0.79 0.05 -1.01 1.8
5 -0.02 -0.5 ...
##  $ gyros_arm_z           : num  -0.18 -0.43 1.13 0.92 -0.54 -0.07 0.89 -0.
69 -0.02 0.79 ...
##  $ accel_arm_x           : int  16 -290 -341 -238 -197 -26 99 -98 -287 -30
1 ...
##  $ accel_arm_y           : int  38 215 245 -57 200 130 79 175 111 -42 ...
##  $ accel_arm_z           : int  93 -90 -87 6 -30 -19 -67 -78 -122 -80 ...
##  $ magnet_arm_x          : int  -326 -325 -264 -173 -170 396 702 535 -367
-420 ...
##  $ magnet_arm_y          : int  385 447 474 257 275 176 15 215 335 294 ...
##  $ magnet_arm_z          : int  481 434 413 633 617 516 217 385 520 49
3 ...
##  $ kurtosis_roll_arm     : logi  NA NA NA NA NA NA ...
##  $ kurtosis_picth_arm    : logi  NA NA NA NA NA NA ...
##  $ kurtosis_yaw_arm      : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_arm     : logi  NA NA NA NA NA NA ...
##  $ skewness_pitch_arm    : logi  NA NA NA NA NA NA ...
##  $ skewness_yaw_arm      : logi  NA NA NA NA NA NA ...
##  $ max_roll_arm          : logi  NA NA NA NA NA NA ...
##  $ max_picth_arm         : logi  NA NA NA NA NA NA ...
##  $ max_yaw_arm           : logi  NA NA NA NA NA NA ...
##  $ min_roll_arm          : logi  NA NA NA NA NA NA ...
```

```
##  $ min_pitch_arm           : logi  NA NA NA NA NA NA ...
##  $ min_yaw_arm             : logi  NA NA NA NA NA NA ...
##  $ amplitude_roll_arm      : logi  NA NA NA NA NA NA ...
##  $ amplitude_pitch_arm     : logi  NA NA NA NA NA NA ...
##  $ amplitude_yaw_arm       : logi  NA NA NA NA NA NA ...
##  $ roll_dumbbell           : num  -17.7 54.5 57.1 43.1 -101.4 ...
##  $ pitch_dumbbell          : num  25 -53.7 -51.4 -30 -53.4 ...
##  $ yaw_dumbbell            : num  126.2 -75.5 -75.2 -103.3 -14.2 ...
##  $ kurtosis_roll_dumbbell  : logi  NA NA NA NA NA NA ...
##  $ kurtosis_picth_dumbbell : logi  NA NA NA NA NA NA ...
##  $ kurtosis_yaw_dumbbell   : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_dumbbell  : logi  NA NA NA NA NA NA ...
##  $ skewness_pitch_dumbbell : logi  NA NA NA NA NA NA ...
##  $ skewness_yaw_dumbbell   : logi  NA NA NA NA NA NA ...
##  $ max_roll_dumbbell       : logi  NA NA NA NA NA NA ...
##  $ max_picth_dumbbell      : logi  NA NA NA NA NA NA ...
##  $ max_yaw_dumbbell        : logi  NA NA NA NA NA NA ...
##  $ min_roll_dumbbell       : logi  NA NA NA NA NA NA ...
##  $ min_pitch_dumbbell      : logi  NA NA NA NA NA NA ...
##  $ min_yaw_dumbbell        : logi  NA NA NA NA NA NA ...
##  $ amplitude_roll_dumbbell : logi  NA NA NA NA NA NA ...
##  $ amplitude_pitch_dumbbell: logi  NA NA NA NA NA NA ...
##  $ amplitude_yaw_dumbbell  : logi  NA NA NA NA NA NA ...
##  $ total_accel_dumbbell    : int  9 31 29 18 4 29 29 29 3 2 ...
##  $ var_accel_dumbbell      : logi  NA NA NA NA NA NA ...
##  $ avg_roll_dumbbell       : logi  NA NA NA NA NA NA ...
##  $ stddev_roll_dumbbell    : logi  NA NA NA NA NA NA ...
##  $ var_roll_dumbbell       : logi  NA NA NA NA NA NA ...
##  $ avg_pitch_dumbbell      : logi  NA NA NA NA NA NA ...
##  $ stddev_pitch_dumbbell   : logi  NA NA NA NA NA NA ...
##  $ var_pitch_dumbbell      : logi  NA NA NA NA NA NA ...
##  $ avg_yaw_dumbbell        : logi  NA NA NA NA NA NA ...
##  $ stddev_yaw_dumbbell     : logi  NA NA NA NA NA NA ...
##  $ var_yaw_dumbbell        : logi  NA NA NA NA NA NA ...
##  $ gyros_dumbbell_x        : num  0.64 0.34 0.39 0.1 0.29 -0.59 0.34 0.37 0.
03 0.42 ...
##  $ gyros_dumbbell_y        : num  0.06 0.05 0.14 -0.02 -0.47 0.8 0.16 0.14
-0.21 0.51 ...
##  $ gyros_dumbbell_z        : num  -0.61 -0.71 -0.34 0.05 -0.46 1.1 -0.23 -0.
39 -0.21 -0.03 ...
##  $ accel_dumbbell_x        : int  21 -153 -141 -51 -18 -138 -145 -140 0
-7 ...
##  $ accel_dumbbell_y        : int  -15 155 155 72 -30 166 150 159 25 -20 ...
##  $ accel_dumbbell_z        : int  81 -205 -196 -148 -5 -186 -190 -191 9
7 ...
##  $ magnet_dumbbell_x       : int  523 -502 -506 -576 -424 -543 -484 -515 -51
9 -531 ...
##  $ magnet_dumbbell_y       : int  -528 388 349 238 252 262 354 350 348 32
1 ...
```

```
##  $ magnet_dumbbell_z       : int  -56 -36 41 53 312 96 97 53 -32 -164 ...
##  $ roll_forearm            : num  141 109 131 0 -176 150 155 -161 15.5 13.
2 ...
##  $ pitch_forearm           : num  49.3 -17.6 -32.6 0 -2.16 1.46 34.5 43.6 -6
3.5 19.4 ...
##  $ yaw_forearm             : num  156 106 93 0 -47.9 89.7 152 -89.5 -139 -10
5 ...
##  $ kurtosis_roll_forearm   : logi  NA NA NA NA NA NA ...
##  $ kurtosis_picth_forearm  : logi  NA NA NA NA NA NA ...
##  $ kurtosis_yaw_forearm    : logi  NA NA NA NA NA NA ...
##  $ skewness_roll_forearm   : logi  NA NA NA NA NA NA ...
##  $ skewness_pitch_forearm  : logi  NA NA NA NA NA NA ...
##  $ skewness_yaw_forearm    : logi  NA NA NA NA NA NA ...
##  $ max_roll_forearm        : logi  NA NA NA NA NA NA ...
##  $ max_picth_forearm       : logi  NA NA NA NA NA NA ...
##  $ max_yaw_forearm         : logi  NA NA NA NA NA NA ...
##  $ min_roll_forearm        : logi  NA NA NA NA NA NA ...
##  $ min_pitch_forearm       : logi  NA NA NA NA NA NA ...
##  $ min_yaw_forearm         : logi  NA NA NA NA NA NA ...
##  $ amplitude_roll_forearm  : logi  NA NA NA NA NA NA ...
##  $ amplitude_pitch_forearm : logi  NA NA NA NA NA NA ...
##  $ amplitude_yaw_forearm   : logi  NA NA NA NA NA NA ...
##  $ total_accel_forearm     : int  33 39 34 43 24 43 32 47 36 24 ...
##  $ var_accel_forearm       : logi  NA NA NA NA NA NA ...
##  $ avg_roll_forearm        : logi  NA NA NA NA NA NA ...
##  $ stddev_roll_forearm     : logi  NA NA NA NA NA NA ...
##  $ var_roll_forearm        : logi  NA NA NA NA NA NA ...
##  $ avg_pitch_forearm       : logi  NA NA NA NA NA NA ...
##  $ stddev_pitch_forearm    : logi  NA NA NA NA NA NA ...
##  $ var_pitch_forearm       : logi  NA NA NA NA NA NA ...
##  $ avg_yaw_forearm         : logi  NA NA NA NA NA NA ...
##  $ stddev_yaw_forearm      : logi  NA NA NA NA NA NA ...
##  $ var_yaw_forearm         : logi  NA NA NA NA NA NA ...
##  $ gyros_forearm_x         : num  0.74 1.12 0.18 1.38 -0.75 -0.88 -0.53 0.6
3 0.03 0.02 ...
##  $ gyros_forearm_y         : num  -3.34 -2.78 -0.79 0.69 3.1 4.26 1.8 -0.74
0.02 0.13 ...
##  $ gyros_forearm_z         : num  -0.59 -0.18 0.28 1.8 0.8 1.35 0.75 0.49
-0.02 -0.07 ...
##  $ accel_forearm_x         : int  -110 212 154 -92 131 230 -192 -151 195 -21
2 ...
##  $ accel_forearm_y         : int  267 297 271 406 -93 322 170 -331 204 9
8 ...
##  $ accel_forearm_z         : int  -149 -118 -129 -39 172 -144 -175 -282 -21
7 -7 ...
##  $ magnet_forearm_x        : int  -714 -237 -51 -233 375 -300 -678 -109 0 -4
03 ...
##  $ magnet_forearm_y        : int  419 791 698 783 -787 800 284 -619 652 72
3 ...
```

```
##  $ magnet_forearm_z        : int  617 873 783 521 91 884 585 -32 469 512 ...
##  $ problem_id              : int  1 2 3 4 5 6 7 8 9 10 ...
```

Check the rows of data which has complete cases Training Dataset

```
sum(complete.cases(RawTrainData)) #406 Very small part of the training data has
complete data
```

```
## [1] 406
```

Test Dataset

```
sum(complete.cases(RawTestData)) #0 None of the test data has complete data
```

```
## [1] 0
```

Data Cleaning and Preperation

We will remove the NAs and irrelevant variables

In the training set check columns with total NA values greater then 10% of the rows (more than 2000 NAs)

```
RawTrainDataNZero <- RawTrainData[, colSums(is.na(RawTrainData)) < 2000]
ncol(RawTrainDataNZero) #93
```

```
## [1] 93
```

93 variables have total number of NAs greater then 10% of the total rows of data

Check the numbers of columns which has one or more NAs

```
RawTrainDataZero <- RawTrainData[, colSums(is.na(RawTrainData)) == 0]
ncol(RawTrainDataZero) #93
```

```
## [1] 93
```

Again 93 columns have no NAs, no need for further detailed processing such as imputing Just remove the columns with NAs

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

In the testing set remove columns with one or more NA in the training dataset

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

Further to this, we need to remove the unnecessary columns that do not contribute to the results

These are: $X:int $user_name:Factor w / 6 levels $raw_timestamp_part_1:int $raw_timestamp_part_2:int $cvtd_timestamp:Factor w / 20 levels $new_window:Factor w / 2 levels $num_window:int

```
classe <- RawTrainData$classe
TrainColsToRemove <- grepl("^X|user_name|timestamp|window", names(RawTrainDat
a))
RawTrainData <- RawTrainData[, !TrainColsToRemove]
CleanTrainData <- RawTrainData[, sapply(RawTrainData, is.numeric)]
CleanTrainData$classe <- classe

classe <- RawTestData$classe
TestColsToRemove <- grepl("^X|user_name|timestamp|window", names(RawTestData))
RawTestData <- RawTestData[, !TestColsToRemove]
CleanTestData <- RawTestData[, sapply(RawTestData, is.numeric)]
CleanTestData$classe <- classe
```

Check the rows of data which has complete cases again Training Dataset

```
sum(complete.cases(RawTrainData)) #19622 We do now have complete cases for all
training data
```

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

Test Dataset

```
sum(complete.cases(RawTestData)) #20 We do now have complete cases for all test
data
```

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

Set seed for reproducible results

```
set.seed(562389)
```

We further the partition the training dataset for training and validataion purposes. We would like to the validate the model with a subset of the data before applying the test data This is to avoid overfitting Generate a training and validation dataset

```
TrainIdx <- createDataPartition(CleanTrainData$classe, p = 0.7, list = FALSE)
TrainData <- CleanTrainData[TrainIdx,]
TestData <- CleanTrainData[ - TrainIdx,]
```

Data Modelling

We will test several algorithms and compare their accuracy levels

First one is the Random Forest

```
modelRf <- randomForest(classe~., data = TrainData)
predRf<-predict(modelRf,TestData)
confusionMatrix(TestData$classe, predRf)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1673    1    0    0    0
##          B    5 1134    0    0    0
##          C    0    6 1017    3    0
##          D    0    0    7  956    1
##          E    0    0    2    1 1079
##
## Overall Statistics
##
##                Accuracy : 0.9956
##                  95% CI : (0.9935, 0.9971)
##     No Information Rate : 0.2851
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9944
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9970   0.9939   0.9912   0.9958   0.9991
## Specificity            0.9998   0.9989   0.9981   0.9984   0.9994
## Pos Pred Value         0.9994   0.9956   0.9912   0.9917   0.9972
## Neg Pred Value         0.9988   0.9985   0.9981   0.9992   0.9998
## Prevalence             0.2851   0.1939   0.1743   0.1631   0.1835
## Detection Rate         0.2843   0.1927   0.1728   0.1624   0.1833
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9984   0.9964   0.9947   0.9971   0.9992
```

Second we test Artificial Neural Network

A simple single layer network with no hidden perceptrons

```
modelNn <- nnet(classe~., data = TrainData, size=15)
```

```
## # weights:  875
## initial  value 31323.559896
## iter  10 value 20121.825391
## iter  20 value 19049.780658
## iter  30 value 18393.909207
## iter  40 value 17720.915949
## iter  50 value 17390.589250
## iter  60 value 17163.768068
## iter  70 value 16934.361590
## iter  80 value 16620.275146
## iter  90 value 16382.999835
## iter 100 value 16237.785398
## final  value 16237.785398
## stopped after 100 iterations
```

```
predNn<-predict(modelNn,TestData, TYPE="class")
```

Output of predictors for Artifical Neural Networks is different Rather then giving the highhest probability prediction It gives the probability for each prediction We select the one with the highest

```
prdNN<-c("A","B","C","D","E")[apply(predNn,1,which.max)]
confusionMatrix(TestData$classe, prdNN)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1069  104  235  227   39
##          B  120  543  146  168  162
##          C  179   94  538  146   69
##          D  106  109   98  531  120
##          E  104  241  187  256  294
##
## Overall Statistics
##
##                Accuracy : 0.5055
##                  95% CI : (0.4927, 0.5184)
##     No Information Rate : 0.2681
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.3771
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.6774  0.49771  0.44684  0.39985  0.42982
## Specificity            0.8595  0.87568  0.89575  0.90498  0.84849
## Pos Pred Value         0.6386  0.47673  0.52437  0.55083  0.27172
## Neg Pred Value         0.8791  0.88453  0.86293  0.83804  0.91880
## Prevalence             0.2681  0.18539  0.20459  0.22566  0.11623
## Detection Rate         0.1816  0.09227  0.09142  0.09023  0.04996
## Detection Prevalence   0.2845  0.19354  0.17434  0.16381  0.18386
## Balanced Accuracy      0.7685  0.68669  0.67130  0.65242  0.63916
```

Third we test support vector algorithm

```
modelSvm <- svm(classe~., data = TrainData)
predSvm<-predict(modelSvm,TestData)
confusionMatrix(TestData$classe, predSvm)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1670    0    2    0    2
##          B   84 1039   16    0    0
##          C    0   30  978   12    6
##          D    6    0   87  870    1
##          E    0   10   24   17 1031
##
## Overall Statistics
##
##                Accuracy : 0.9495
##                  95% CI : (0.9436, 0.955)
##     No Information Rate : 0.2991
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.936
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9489   0.9629   0.8835   0.9677   0.9913
## Specificity            0.9990   0.9792   0.9900   0.9811   0.9895
## Pos Pred Value         0.9976   0.9122   0.9532   0.9025   0.9529
## Neg Pred Value         0.9786   0.9916   0.9735   0.9941   0.9981
## Prevalence             0.2991   0.1833   0.1881   0.1528   0.1767
## Detection Rate         0.2838   0.1766   0.1662   0.1478   0.1752
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9739   0.9711   0.9367   0.9744   0.9904
```

Fourth we would like to test Linear Discriminant Analysis for its simplicity

```
modelLda <- lda(classe~., data = TrainData)
predLda<-predict(modelLda,TestData)
confusionMatrix(TestData$classe, predLda$class)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1360   34  140  137    3
##          B  174  731  144   36   54
##          C   99  107  669  115   36
##          D   56   47  109  708   44
##          E   31  170   95  103  683
##
## Overall Statistics
##
##                Accuracy : 0.7054
##                  95% CI : (0.6935, 0.717)
##     No Information Rate : 0.2923
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6273
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7907   0.6713   0.5782   0.6442   0.8329
## Specificity            0.9246   0.9149   0.9245   0.9465   0.9212
## Pos Pred Value         0.8124   0.6418   0.6520   0.7344   0.6312
## Neg Pred Value         0.9145   0.9246   0.8996   0.9205   0.9715
## Prevalence             0.2923   0.1850   0.1966   0.1867   0.1393
## Detection Rate         0.2311   0.1242   0.1137   0.1203   0.1161
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.8577   0.7931   0.7514   0.7954   0.8771
```

Note that the output of predictor is also different in this case, we need to select the class variable

Fifth and last one is the regression tree

Which are useful for their human interpretable results

```
modelCart <- rpart(classe~., data = TrainData)
predCart<-predict(modelCart,TestData)
```

Output of predictors for Classification and Regression Tree is different Rather then giving the highhest probability prediction It gives the probability for each prediction We select the one with the highest

```
prdCART<-c("A","B","C","D","E")[apply(predCart,1,which.max)]
confusionMatrix(TestData$classe, prdCART)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1510   38   66   42   18
##          B  256  694  103   58   28
##          C   27  144  794   53    8
##          D   96   43  179  597   49
##          E   35   54   91   70  832
##
## Overall Statistics
##
##                Accuracy : 0.7523
##                  95% CI : (0.741, 0.7632)
##     No Information Rate : 0.3269
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.685
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.7848   0.7133   0.6440   0.7280   0.8898
## Specificity            0.9586   0.9094   0.9501   0.9275   0.9495
## Pos Pred Value         0.9020   0.6093   0.7739   0.6193   0.7689
## Neg Pred Value         0.9017   0.9412   0.9097   0.9547   0.9786
## Prevalence             0.3269   0.1653   0.2095   0.1393   0.1589
## Detection Rate         0.2566   0.1179   0.1349   0.1014   0.1414
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.8717   0.8113   0.7970   0.8278   0.9197
```

Results of the comparision of accuracy of the different algorithms

```
OutputTable <- c("Random Forest",as.numeric(confusionMatrix(TestData$classe, pr
edRf)$overall["Accuracy"]))
temp<- rbind(OutputTable, c("Artificial Neural Network",as.numeric(confusionMat
rix(TestData$classe, prdNN)$overall["Accuracy"])))
temp<- rbind(temp, c("Support Vector",as.numeric(confusionMatrix(TestData$class
e, predSvm)$overall["Accuracy"])))
temp<- rbind(temp, c("Linear Discriminant Analysis",as.numeric(confusionMatrix
(TestData$classe, predLda$class)$overall["Accuracy"])))
temp<- rbind(temp, c("Classification and Regression Tree",as.numeric(confusionM
atrix(TestData$classe, prdCART)$overall["Accuracy"])))
OutputTable <- temp
colnames(OutputTable) <- c("Algorithm", "Accuracy")
OutputTable
```

```
##             Algorithm                          Accuracy
## OutputTable "Random Forest"                    "0.995581988105353"
##             "Artificial Neural Network"        "0.505522514868309"
##             "Support Vector"                   "0.949532710280374"
##             "Linear Discriminant Analysis"     "0.7053525913339"
##             "Classification and Regression Tree" "0.752251486830926"
```

By far random forest is the most acurate one We will use this algorithm for predicting the outcome for the course project

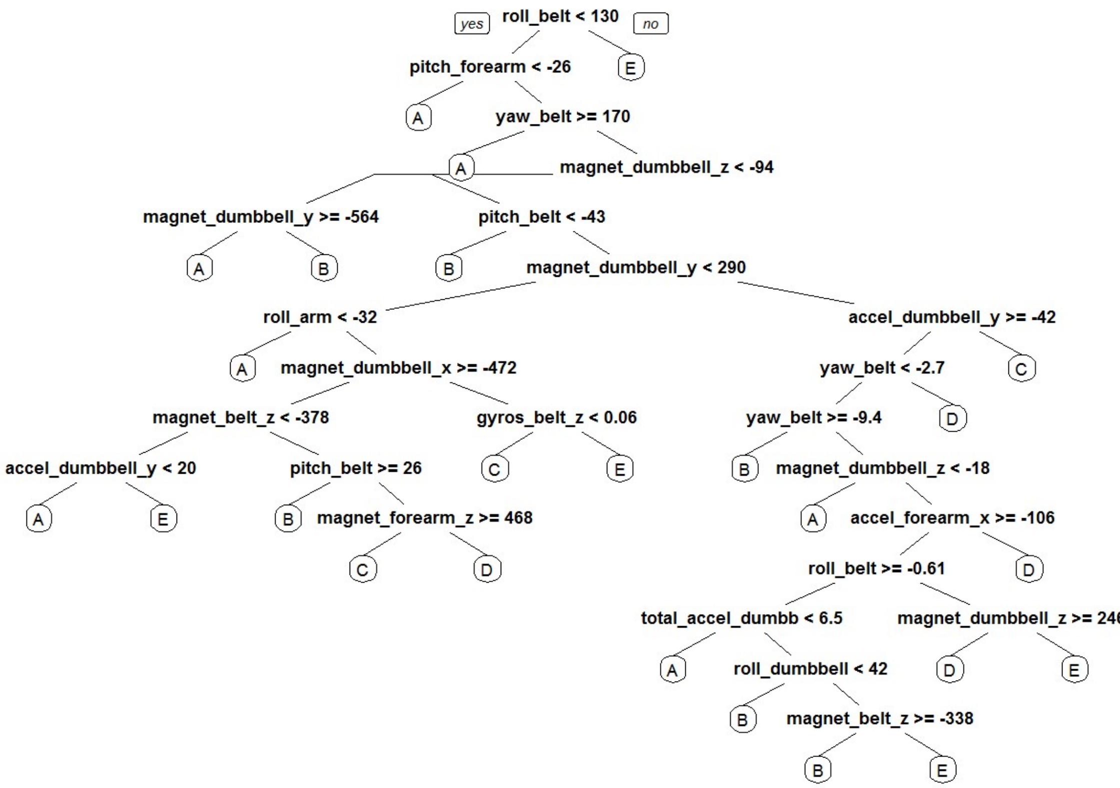
```
predRf<-predict(modelRf,CleanTestData)
predRf
```

```
##  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
```

Visual Analysis of the Model

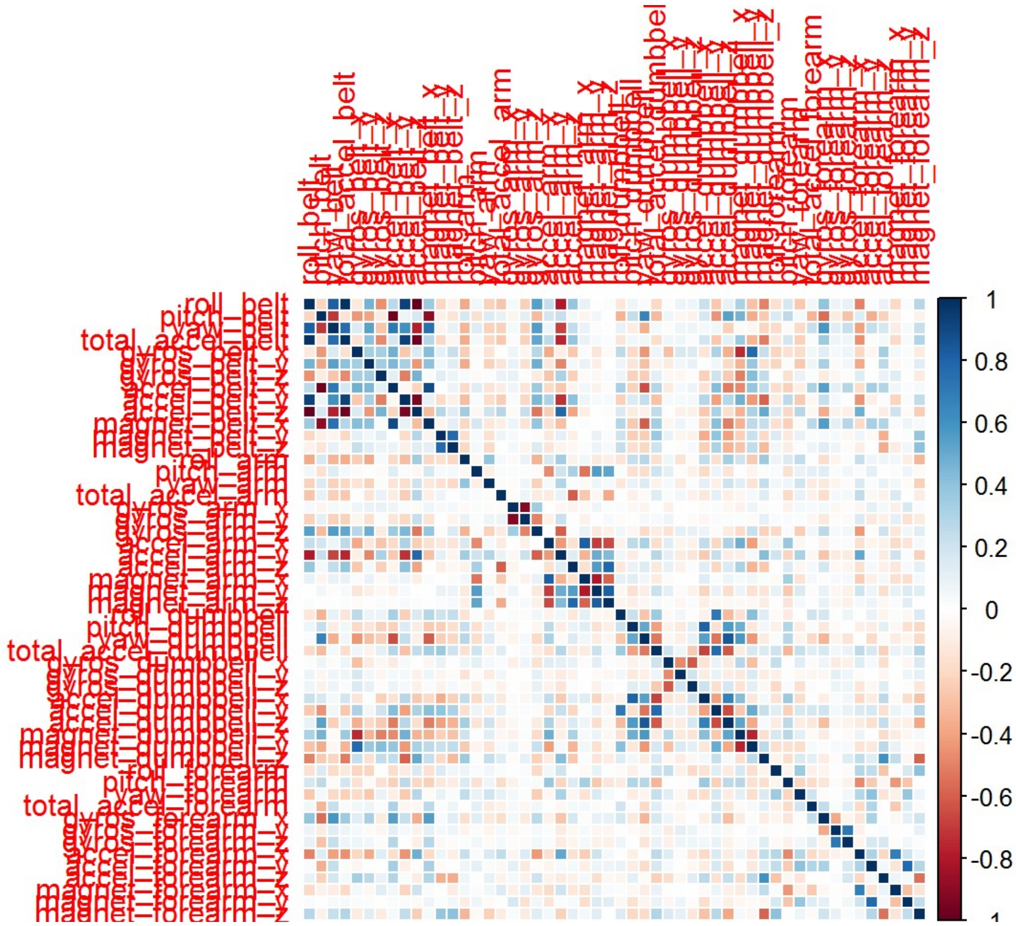
An additional tree visualization would reveal the Decision critieria Especialy important to be analyzed by a domain expert Sometime data driven models could bring some issues

```
treeModel <- rpart(classe ~ ., data=TrainData, method="class")
prp(treeModel)
```



Correlation Plot between different attributes of the model The graph is not readable but gives an idea on the importance of some of the attributes of the model

```
corrPlot <- cor(TrainData[, -length(names(TrainData))])
corrplot(corrPlot, method="color")
```


roll_belt
pitch_belt
total_accel_belt
roll_arm
pitch_arm
total_accel_arm
pitch_dumbbell
total_accel_dumbbell
pitch_forearm
total_accel_forearm
magnet_forearm_z
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8