An accessment : How do you know you exercise correctly ?

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 (see the section on the Weight Lifting Exercise Dataset).

Library

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.2.2
```

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

```
## Warning: package 'ggplot2' was built under R version 3.2.2
```

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

```
## Warning: package 'rpart.plot' was built under R version 3.2.2
```

```
library(RColorBrewer)
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.2.2
```

```
## Loading required package: RGtk2
```

```
## Warning: package 'RGtk2' was built under R version 3.2.2
```

```
## 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.
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.2.2
```

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
```

Random Number Generation

Integer vector, containing the random number generator (RNG) state for random number generation in R

```
set.seed(11111)
```

Dataset

The training data for this project are available here:

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

The test data are available here:

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

The data for this project come from this source: http://groupware.les.inf.puc-rio.br/har.

Download both training dataset :-

```
curdir <-getwd()
file.url<-'http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv'
download.file(file.url,destfile=paste(curdir,'/pml-training.csv',sep=""))

curdir <-getwd()
file.url<-'http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv'
download.file(file.url,destfile=paste(curdir,'/pml-testing.csv',sep=""))
```

Load both dataset :-

```
train <- read.csv(paste(curdir,'/pml-training.csv',sep=""),na.strings=c("NA","#DIV/0!",""))
test <-read.csv(paste(curdir,'/pml-testing.csv',sep=""),na.strings=c("NA","#DIV/0!",""))
```

Cheking the dimension of training and test dataset :-

```
dim(train)
```

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

```
dim(test)
```

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

Checking the columns which have all missing values

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

We remove 6 of the variables which is irrelevant like :-

a) user_name

b) raw_timestamp_part_1

c) raw_timestamp_part_2

d) cvtd_timestamp

e) new_window

f) num_window

which resides on the column 1-7.

```
train <-train[,-c(1:7)]
test <-test[,-c(1:7)]
```

Check again the dimension

```
dim(train)
```

```
## [1] 19622    53
```

```
dim(test)
```

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

Now we obtain the several rows to preview

```
head(train)
```

```
##   roll_belt pitch_belt yaw_belt total_accel_belt gyros_belt_x gyros_belt_y
## 1      1.41       8.07    -94.4                3         0.00         0.00
## 2      1.41       8.07    -94.4                3         0.02         0.00
## 3      1.42       8.07    -94.4                3         0.00         0.00
## 4      1.48       8.05    -94.4                3         0.02         0.00
## 5      1.48       8.07    -94.4                3         0.02         0.02
## 6      1.45       8.06    -94.4                3         0.02         0.00
##   gyros_belt_z accel_belt_x accel_belt_y accel_belt_z magnet_belt_x
## 1        -0.02          -21            4           22            -3
## 2        -0.02          -22            4           22            -7
## 3        -0.02          -20            5           23            -2
## 4        -0.03          -22            3           21            -6
## 5        -0.02          -21            2           24            -6
## 6        -0.02          -21            4           21             0
##   magnet_belt_y magnet_belt_z roll_arm pitch_arm yaw_arm total_accel_arm
## 1           599          -313     -128      22.5    -161              34
```

```
## 2           608          -311     -128      22.5    -161              34
## 3           600          -305     -128      22.5    -161              34
## 4           604          -310     -128      22.1    -161              34
## 5           600          -302     -128      22.1    -161              34
## 6           603          -312     -128      22.0    -161              34
##   gyros_arm_x gyros_arm_y gyros_arm_z accel_arm_x accel_arm_y accel_arm_z
## 1        0.00        0.00       -0.02        -288         109        -123
## 2        0.02       -0.02       -0.02        -290         110        -125
## 3        0.02       -0.02       -0.02        -289         110        -126
## 4        0.02       -0.03        0.02        -289         111        -123
## 5        0.00       -0.03        0.00        -289         111        -123
## 6        0.02       -0.03        0.00        -289         111        -122
##   magnet_arm_x magnet_arm_y magnet_arm_z roll_dumbbell pitch_dumbbell
## 1         -368          337          516      13.05217      -70.49400
## 2         -369          337          513      13.13074      -70.63751
## 3         -368          344          513      12.85075      -70.27812
## 4         -372          344          512      13.43120      -70.39379
## 5         -374          337          506      13.37872      -70.42856
## 6         -369          342          513      13.38246      -70.81759
##   yaw_dumbbell total_accel_dumbbell gyros_dumbbell_x gyros_dumbbell_y
## 1    -84.87394                   37                0            -0.02
## 2    -84.71065                   37                0            -0.02
## 3    -85.14078                   37                0            -0.02
## 4    -84.87363                   37                0            -0.02
## 5    -84.85306                   37                0            -0.02
## 6    -84.46500                   37                0            -0.02
##   gyros_dumbbell_z accel_dumbbell_x accel_dumbbell_y accel_dumbbell_z
## 1             0.00             -234               47             -271
## 2             0.00             -233               47             -269
## 3             0.00             -232               46             -270
## 4            -0.02             -232               48             -269
## 5             0.00             -233               48             -270
## 6             0.00             -234               48             -269
##   magnet_dumbbell_x magnet_dumbbell_y magnet_dumbbell_z roll_forearm
## 1              -559               293               -65         28.4
## 2              -555               296               -64         28.3
## 3              -561               298               -63         28.3
## 4              -552               303               -60         28.1
## 5              -554               292               -68         28.0
## 6              -558               294               -66         27.9
##   pitch_forearm yaw_forearm total_accel_forearm gyros_forearm_x
## 1         -63.9        -153                 36            0.03
## 2         -63.9        -153                 36            0.02
## 3         -63.9        -152                 36            0.03
## 4         -63.9        -152                 36            0.02
## 5         -63.9        -152                 36            0.02
## 6         -63.9        -152                 36            0.02
##   gyros_forearm_y gyros_forearm_z accel_forearm_x accel_forearm_y
## 1            0.00           -0.02             192             203
## 2            0.00           -0.02             192             203
## 3           -0.02            0.00             196             204
## 4           -0.02            0.00             189             206
## 5            0.00           -0.02             189             206
## 6           -0.02           -0.03             193             203
```

```
##   accel_forearm_z magnet_forearm_x magnet_forearm_y magnet_forearm_z
## 1            -215              -17              654              476
## 2            -216              -18              661              473
## 3            -213              -18              658              469
## 4            -214              -16              658              469
## 5            -214              -17              655              473
## 6            -215               -9              660              478
##   classe
## 1      A
## 2      A
## 3      A
## 4      A
## 5      A
## 6      A
```

```
head(test)
```

```
##   roll_belt pitch_belt yaw_belt total_accel_belt gyros_belt_x gyros_belt_y
## 1    123.00      27.00    -4.75               20        -0.50        -0.02
## 2      1.02       4.87   -88.90                4        -0.06        -0.02
## 3      0.87       1.82   -88.50                5         0.05         0.02
## 4    125.00     -41.60   162.00               17         0.11         0.11
## 5      1.35       3.33   -88.60                3         0.03         0.02
## 6     -5.92       1.59   -87.70                4         0.10         0.05
##   gyros_belt_z accel_belt_x accel_belt_y accel_belt_z magnet_belt_x
## 1        -0.46          -38           69         -179           -13
## 2        -0.07          -13           11           39            43
## 3         0.03            1           -1           49            29
## 4        -0.16           46           45         -156           169
## 5         0.00           -8            4           27            33
## 6        -0.13          -11          -16           38            31
##   magnet_belt_y magnet_belt_z roll_arm pitch_arm yaw_arm total_accel_arm
## 1           581          -382     40.7    -27.80     178              10
## 2           636          -309      0.0      0.00       0              38
## 3           631          -312      0.0      0.00       0              44
## 4           608          -304   -109.0     55.00    -142              25
## 5           566          -418     76.1      2.76     102              29
## 6           638          -291      0.0      0.00       0              14
##   gyros_arm_x gyros_arm_y gyros_arm_z accel_arm_x accel_arm_y accel_arm_z
## 1       -1.65        0.48       -0.18          16          38          93
## 2       -1.17        0.85       -0.43        -290         215         -90
## 3        2.10       -1.36        1.13        -341         245         -87
## 4        0.22       -0.51        0.92        -238         -57           6
## 5       -1.96        0.79       -0.54        -197         200         -30
## 6        0.02        0.05       -0.07         -26         130         -19
##   magnet_arm_x magnet_arm_y magnet_arm_z roll_dumbbell pitch_dumbbell
## 1         -326          385          481     -17.73748       24.96085
## 2         -325          447          434      54.47761      -53.69758
## 3         -264          474          413      57.07031      -51.37303
## 4         -173          257          633      43.10927      -30.04885
## 5         -170          275          617    -101.38396      -53.43952
## 6          396          176          516      62.18750      -50.55595
##   yaw_dumbbell total_accel_dumbbell gyros_dumbbell_x gyros_dumbbell_y
## 1    126.23596                   9             0.64             0.06
```

```
## 2    -75.51480                   31             0.34             0.05
## 3    -75.20287                   29             0.39             0.14
## 4   -103.32003                   18             0.10            -0.02
## 5    -14.19542                    4             0.29            -0.47
## 6    -71.12063                   29            -0.59             0.80
##   gyros_dumbbell_z accel_dumbbell_x accel_dumbbell_y accel_dumbbell_z
## 1            -0.61               21              -15               81
## 2            -0.71             -153              155             -205
## 3            -0.34             -141              155             -196
## 4             0.05              -51               72             -148
## 5            -0.46              -18              -30               -5
## 6             1.10             -138              166             -186
##   magnet_dumbbell_x magnet_dumbbell_y magnet_dumbbell_z roll_forearm
## 1               523              -528               -56          141
## 2              -502               388               -36          109
## 3              -506               349                41          131
## 4              -576               238                53            0
## 5              -424               252               312         -176
## 6              -543               262                96          150
##   pitch_forearm yaw_forearm total_accel_forearm gyros_forearm_x
## 1         49.30       156.0                 33            0.74
## 2        -17.60       106.0                 39            1.12
## 3        -32.60        93.0                 34            0.18
## 4          0.00         0.0                 43            1.38
## 5         -2.16       -47.9                 24           -0.75
## 6          1.46        89.7                 43           -0.88
##   gyros_forearm_y gyros_forearm_z accel_forearm_x accel_forearm_y
## 1           -3.34           -0.59            -110             267
## 2           -2.78           -0.18             212             297
## 3           -0.79            0.28             154             271
## 4            0.69            1.80             -92             406
## 5            3.10            0.80             131             -93
## 6            4.26            1.35             230             322
##   accel_forearm_z magnet_forearm_x magnet_forearm_y magnet_forearm_z
## 1            -149             -714              419              617
## 2            -118             -237              791              873
## 3            -129              -51              698              783
## 4             -39             -233              783              521
## 5             172              375             -787               91
## 6            -144             -300              800              884
##   problem_id
## 1          1
## 2          2
## 3          3
## 4          4
## 5          5
## 6          6
```

In order to run cross-validation , the training dataset need to partition into 2 sets . We set the 1st partition for training dataset to 75% and test dataset to 25%. Training dataset contains 53 variables with 19622 obs and test dataset contains 53 variables with 20 obs.

This will do the randomize sub-sampling without replacement

```
PartTrain <- createDataPartition(y=train$classe, p=0.75, list=FALSE)
train_part <- train[PartTrain, ];
test_part <- train[-PartTrain, ]
dim(train_part)
```

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

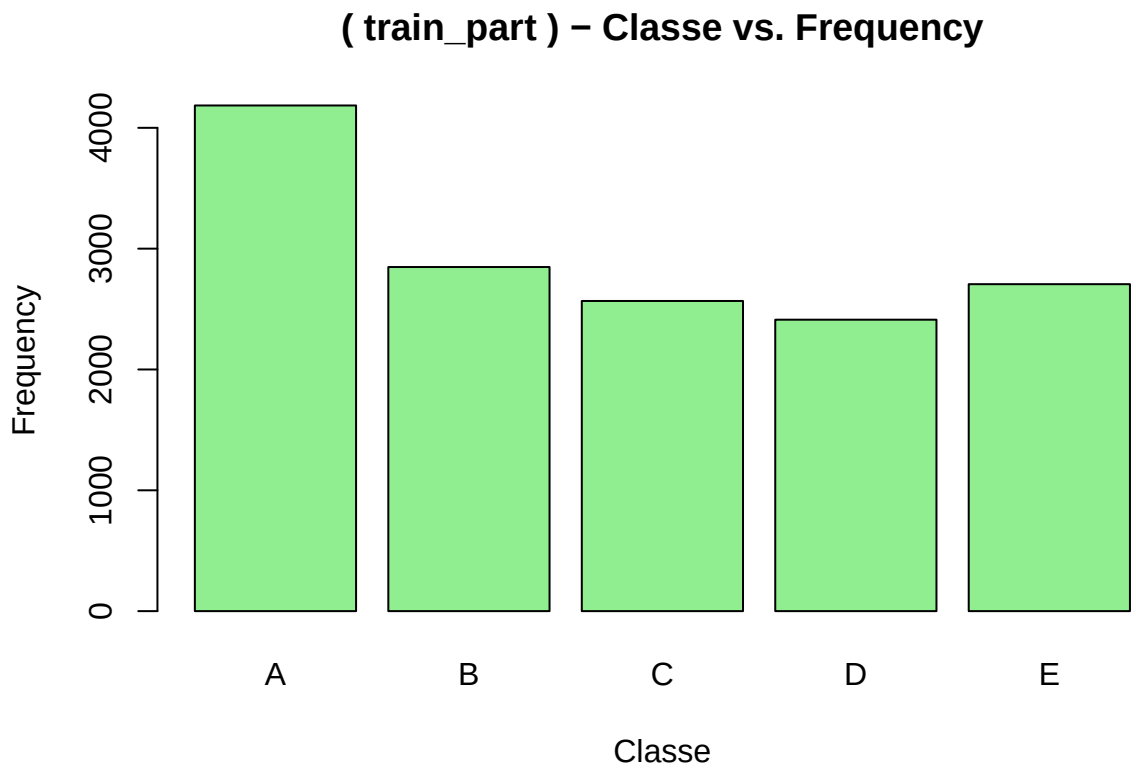
```
dim(test_part)
```

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

Visualization

We try to plot into the histogram to see the trending frequency of each sub-training & test dataset by comparing with each other. The variable classe contains 5 levels which is A,B,C,D & E

```
plot(train_part$classe, col="lightgreen",
main="( train_part ) - Classe vs. Frequency",
xlab="Classe", ylab="Frequency")
```



The graph above shows that A ~ 4000x occurrences is most frequent while D is the lest frequent ~ 2500x occurrences

Decision Tree

Decision Tree machine learning algorithm as a support tool that uses a tree-like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility.

```
fit_model <- rpart(classe ~ ., data=train_part, method="class")
fit_model
```

```
## n= 14718
##
## node), split, n, loss, yval, (yprob)
##       * denotes terminal node
##
##     1) root 14718 10533 A (0.28 0.19 0.17 0.16 0.18)
##       2) roll_belt< 130.5 13494  9321 A (0.31 0.21 0.19 0.18 0.11)
##         4) pitch_forearm< -33.65 1196    11 A (0.99 0.0092 0 0 0) *
##         5) pitch_forearm>=-33.65 12298  9310 A (0.24 0.23 0.21 0.2 0.12)
##          10) magnet_dumbbell_y< 438.5 10412  7485 A (0.28 0.18 0.24 0.19 0.11)
##            20) roll_forearm< 123.5 6489  3845 A (0.41 0.18 0.18 0.17 0.06)
##              40) magnet_dumbbell_z< -27.5 2175   717 A (0.67 0.21 0.011 0.077 0.031)
##                80) roll_forearm>=-136.5 1805   393 A (0.78 0.17 0.011 0.028 0.0078) *
##                81) roll_forearm< -136.5 370   218 B (0.12 0.41 0.0081 0.31 0.14) *
##              41) magnet_dumbbell_z>=-27.5 4314  3128 A (0.27 0.16 0.27 0.21 0.075)
##                82) accel_dumbbell_y>=-40.5 3792  2610 A (0.31 0.18 0.19 0.24 0.08)
##                 164) yaw_belt>=169.5 525    69 A (0.87 0.063 0 0.069 0) *
##                 165) yaw_belt< 169.5 3267  2394 D (0.22 0.2 0.22 0.27 0.093)
##                   330) pitch_belt< -42.95 376    69 B (0.021 0.82 0.09 0.048 0.024) *
##                   331) pitch_belt>=-42.95 2891  2036 D (0.25 0.12 0.23 0.3 0.1)
##                     662) roll_belt>=125.5 692   286 C (0.37 0.026 0.59 0.01 0.0043)
##                      1324) magnet_belt_z< -322.5 232     9 A (0.96 0 0.026 0 0.013) *
##                      1325) magnet_belt_z>=-322.5 460    60 C (0.076 0.039 0.87 0.015 0) *
##                     663) roll_belt< 125.5 2199  1351 D (0.21 0.15 0.12 0.39 0.13)
##                      1326) yaw_belt< -85.55 1366  1032 A (0.24 0.22 0.14 0.22 0.19)
##                        2652) accel_dumbbell_z< 21.5 797   486 A (0.39 0.14 0.23 0.22 0.023)
##                          5304) yaw_forearm>=-94.9 608   297 A (0.51 0.18 0.24 0.048 0.021)
##                            10608) magnet_forearm_z>=-151.5 393    87 A (0.78 0.15 0.015 0.025 0.028) *
##                            10609) magnet_forearm_z< -151.5 215    78 C (0.023 0.24 0.64 0.088 0.0093)
##                          5305) yaw_forearm< -94.9 189    45 D (0 0.016 0.2 0.76 0.026) *
##                        2653) accel_dumbbell_z>=21.5 569   330 E (0.04 0.32 0.011 0.21 0.42)
##                          5306) roll_dumbbell< 38.68377 189    45 B (0.026 0.76 0.032 0.032 0.15) *
##                          5307) roll_dumbbell>=38.68377 380   169 E (0.047 0.095 0 0.3 0.56) *
##                      1327) yaw_belt>=-85.55 833   279 D (0.15 0.038 0.1 0.67 0.042)
##                        2654) yaw_arm< -104.85 114     0 A (1 0 0 0 0) *
##                        2655) yaw_arm>=-104.85 719   165 D (0.017 0.045 0.12 0.77 0.049) *
##                82) accel_dumbbell_y< -40.5 522    63 C (0.0077 0.042 0.88 0.033 0.038) *
##            21) roll_forearm>=123.5 3923  2622 C (0.072 0.18 0.33 0.23 0.19)
##              42) accel_forearm_x>=-108.5 2774  1730 C (0.08 0.22 0.38 0.11 0.22)
##                84) magnet_dumbbell_z>=286.5 679   481 A (0.29 0.28 0.016 0.15 0.26)
##                 168) accel_dumbbell_z< 36 230    46 A (0.8 0.091 0.043 0.065 0) *
##                 169) accel_dumbbell_z>=36 449   274 E (0.031 0.38 0.0022 0.2 0.39)
##                   338) roll_dumbbell< 41.74938 184    29 B (0.043 0.84 0.0054 0.033 0.076) *
##                   339) roll_dumbbell>=41.74938 265   104 E (0.023 0.06 0 0.31 0.61) *
##                85) magnet_dumbbell_z< 286.5 2095  1062 C (0.012 0.2 0.49 0.092 0.21)
```

```
##                 170) roll_belt>=-0.255 1917   886 C (0.013 0.21 0.54 0.083 0.16) *
##                 171) roll_belt< -0.255 178    46 E (0 0.062 0.011 0.19 0.74) *
##               43) accel_forearm_x< -108.5 1149   535 D (0.052 0.084 0.22 0.53 0.11)
##                 86) magnet_arm_x< 235.5 377   166 C (0.029 0.15 0.56 0.19 0.064) *
##                 87) magnet_arm_x>=235.5 772   231 D (0.063 0.049 0.06 0.7 0.13) *
##           11) magnet_dumbbell_y>=438.5 1886   919 B (0.032 0.51 0.038 0.22 0.2)
##             22) total_accel_dumbbell>=5.5 1361   477 B (0.045 0.65 0.052 0.021 0.23)
##               44) roll_belt>=-0.575 1140   256 B (0.054 0.78 0.062 0.025 0.084) *
##               45) roll_belt< -0.575 221     0 E (0 0 0 0 1) *
##             23) total_accel_dumbbell< 5.5 525   143 D (0 0.16 0.0019 0.73 0.11) *
##       3) roll_belt>=130.5 1224    12 E (0.0098 0 0 0 0.99) *
```

Displays the (Complexity) cp table for fitted model .

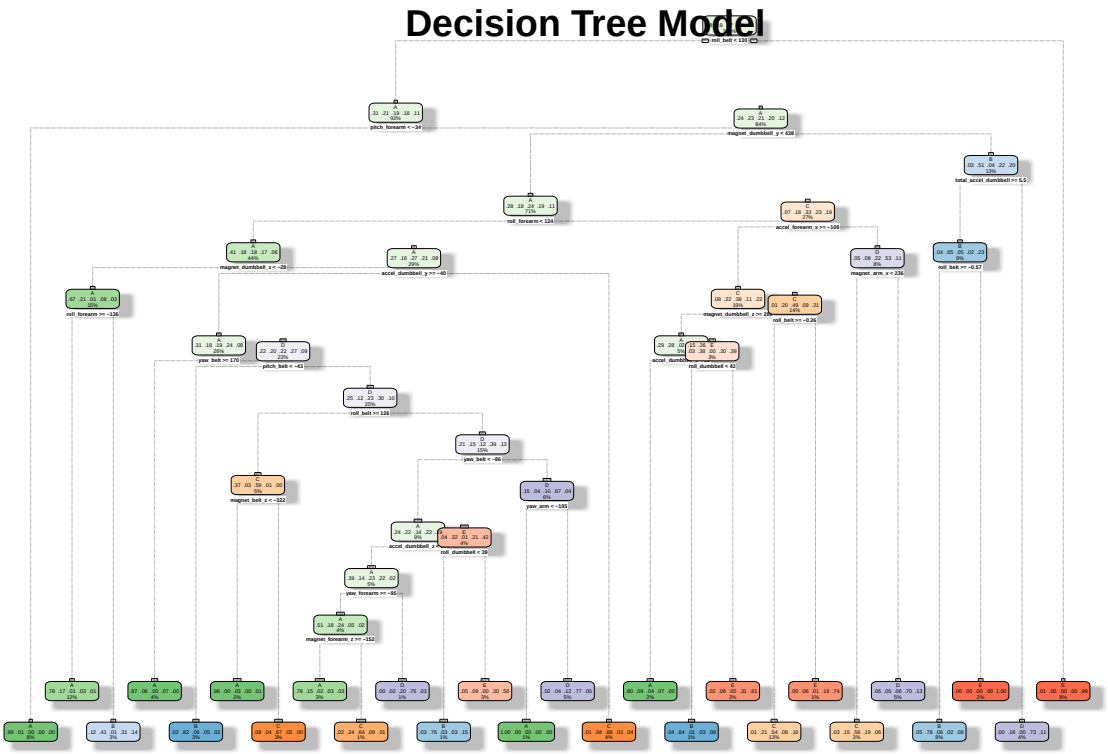
```
printcp(fit_model)
```

```
##
## Classification tree:
## rpart(formula = classe ~ ., data = train_part, method = "class")
##
## Variables actually used in tree construction:
##  [1] accel_dumbbell_y     accel_dumbbell_z     accel_forearm_x
##  [4] magnet_arm_x         magnet_belt_z        magnet_dumbbell_y
##  [7] magnet_dumbbell_z    magnet_forearm_z     pitch_belt
## [10] pitch_forearm        roll_belt            roll_dumbbell
## [13] roll_forearm         total_accel_dumbbell yaw_arm
## [16] yaw_belt             yaw_forearm
##
## Root node error: 10533/14718 = 0.71565
##
## n= 14718
##
##          CP nsplit rel error  xerror      xstd
## 1  0.113928      0   1.00000 1.00000 0.0051957
## 2  0.060888      1   0.88607 0.88664 0.0055466
## 3  0.033893      4   0.70341 0.70597 0.0057586
## 4  0.028387      5   0.66951 0.67388 0.0057553
## 5  0.021599      6   0.64113 0.64521 0.0057421
## 6  0.020982     11   0.51866 0.52815 0.0055848
## 7  0.020602     12   0.49767 0.51078 0.0055468
## 8  0.017754     13   0.47707 0.48334 0.0054786
## 9  0.015285     14   0.45932 0.45191 0.0053878
## 10 0.013386     15   0.44403 0.44261 0.0053582
## 11 0.013102     16   0.43065 0.41460 0.0052615
## 12 0.012342     17   0.41754 0.40587 0.0052288
## 13 0.012152     18   0.40520 0.39960 0.0052047
## 14 0.011013     22   0.35469 0.37596 0.0051078
## 15 0.010823     23   0.34368 0.36030 0.0050385
## 16 0.010064     24   0.33286 0.34786 0.0049804
## 17 0.010000     25   0.32280 0.34530 0.0049680
```

To visualize the decision tree , we use this fancyRpartPlot command below :-

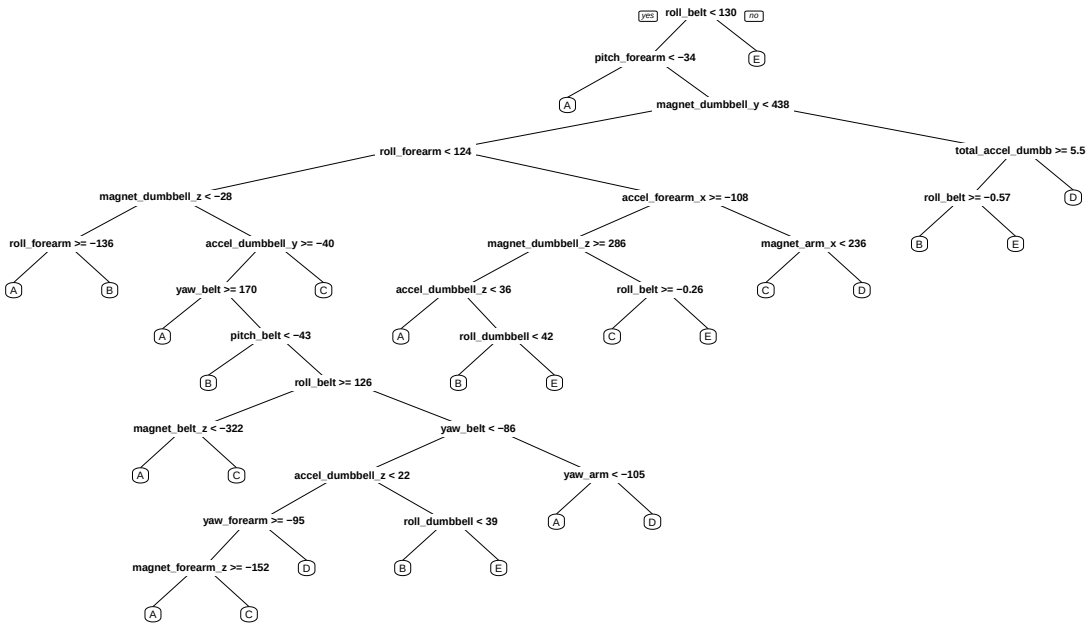
```
fancyRpartPlot(fit_model,main="Decision Tree Model")
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



Green nodes represent individuals classified by the tree as A, blue nodes are those classified as B and orange nodes are classified as C. The gradient is a visual representation of the three numbers in the middle of the nodes: bearing in mind that levels of a factor are by default in alphabetical order, the first of these three numbers is the proportion of individuals in that node that were actually classified as the first level, (A), in train_part ; the second number is the proportion that were actually classified as B, and the third the proportion that were C.

```
rpart.plot(fit_model,main="Classification Tree Model", under=TRUE, faclen=1)
```



Now we predict the fit model for test dataset .

```
model_predictions <- predict(fit_model, test_part, type = "class")
```

Confusion Matrix

Confusion matrix, also known as a contingency table or an error matrix , is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one (in unsupervised learning it is usually called a matching matrix). Each column of the matrix represents the instances in a predicted class while each row represents the instances in an actual class (or vice-versa).

```
confusionMatrix(model_predictions, test_part$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1288  138   23   33    5
##          B   43  550   60   66   78
##          C   31  174  711   88  116
##          D   24   68   60  541   61
##          E    9   19    1   76  641
##
## Overall Statistics
```

```
##
##                Accuracy : 0.7608
##                  95% CI : (0.7486, 0.7727)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.697
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9233   0.5796   0.8316   0.6729   0.7114
## Specificity            0.9433   0.9375   0.8990   0.9480   0.9738
## Pos Pred Value         0.8662   0.6901   0.6348   0.7175   0.8592
## Neg Pred Value         0.9687   0.9028   0.9619   0.9366   0.9375
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2626   0.1122   0.1450   0.1103   0.1307
## Detection Prevalence   0.3032   0.1625   0.2284   0.1538   0.1521
## Balanced Accuracy      0.9333   0.7586   0.8653   0.8105   0.8426
```

Random Forest

Random forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. Random forests correct for decision trees' habit of overfitting to their training set.

```
fit_model2 <- randomForest(classe ~. , data=train_part)
```

Now we predict the fit model for test dataset .

```
model_predictions2 <- predict(fit_model2, test_part, type = "class")
```

Below is the confusion matrix of the test results

```
confusionMatrix(model_predictions2, test_part$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    5    0    0    0
##          B    0  942    2    0    0
##          C    0    2  852    5    0
##          D    0    0    1  796    0
##          E    0    0    0    3  901
##
## Overall Statistics
##
##                Accuracy : 0.9963
```

```
##                  95% CI : (0.9942, 0.9978)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9954
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9926   0.9965   0.9900   1.0000
## Specificity            0.9986   0.9995   0.9983   0.9998   0.9993
## Pos Pred Value         0.9964   0.9979   0.9919   0.9987   0.9967
## Neg Pred Value         1.0000   0.9982   0.9993   0.9981   1.0000
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1921   0.1737   0.1623   0.1837
## Detection Prevalence   0.2855   0.1925   0.1752   0.1625   0.1843
## Balanced Accuracy      0.9993   0.9961   0.9974   0.9949   0.9996
```

Conclusion

From the machine learning method above , the accuracy of the Decision Tree is 75.9% and the Random Forest is 99.6% which is better.

Random Forests generally needs larger number of instances to work its randomization concept well and generalize to the novel data. In addition, in one way or another, random forests works with combination of some kind of soft linear boundaries at the decision surface

Prediction files generator for assignment submission code

```
n=0
x=0
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(model_predictions2)
```

Reference

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. Read more: http://groupware.les.inf.puc-rio.br/har#ixzz3lj0hACeI