How much activity?

Fernando Roque

13 de agosto de 2016

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).

Data

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]. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

Exploratory Analisys

The first step is load the data prevuiuosly downloaded from the web and explore the data.

```
setwd("C:/Users/Fernando/Documents/Material/Coursera/DataScience/Pracrical machine")

# Loading data
train_data <- read.csv("pml-training.csv", na.strings=c("NA","#DIV/0!",""))
rtest_data  <- read.csv("pml-testing.csv",  na.strings=c("NA","#DIV/0!",""))
```

```
dim(train_data)
```

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

```
dim(rtest_data)
```

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

If we explore the data, we will see that there is a lot of NA's. We can observe that variables with a lot of NA's correspond to summary statistics such as kurtosis, max, skewness, etc. new_Window variable help us in determining if the record correspon to a summary statistics, i.e., `yes` value indicates a summary statistics record. The following table summarize how much of this record exists:

```r
table(train_data$new_window)
```

```
##
##    no   yes
## 19216   406
```

Above table indicates that there is 406 records taht we can deleted from the analysis. The following code delete this observations.

```r
trainc <- train_data[train_data$new_window!="yes",]
```

If we drop this records, summary statistics variables will have only NA's. To eliminate this variables, we will use grep function to reference variable with an especific character inicial name.

```r
ncol <- data.frame(ncol=names(trainc))
kur <- with(ncol, grep("kurtosis",ncol))
max <- with(ncol, grep("max",ncol))
min <- with(ncol, grep("min",ncol))
ske <- with(ncol, grep("skewness",ncol))
amp <- with(ncol, grep("amplitude",ncol))
var <- with(ncol, grep("var",ncol))
avg <- with(ncol, grep("avg",ncol))
std <- with(ncol, grep("stddev",ncol))
oth <- c(1:7) # Other general useless variables
nvar<-c(oth, kur, max, min, ske, amp, var, avg, std)
traincv <- trainc[,-nvar]
testcv <- rtest_data[,-nvar]
```

The following are the variables we will omit in the analisys. The resulting is storage in `traincv`

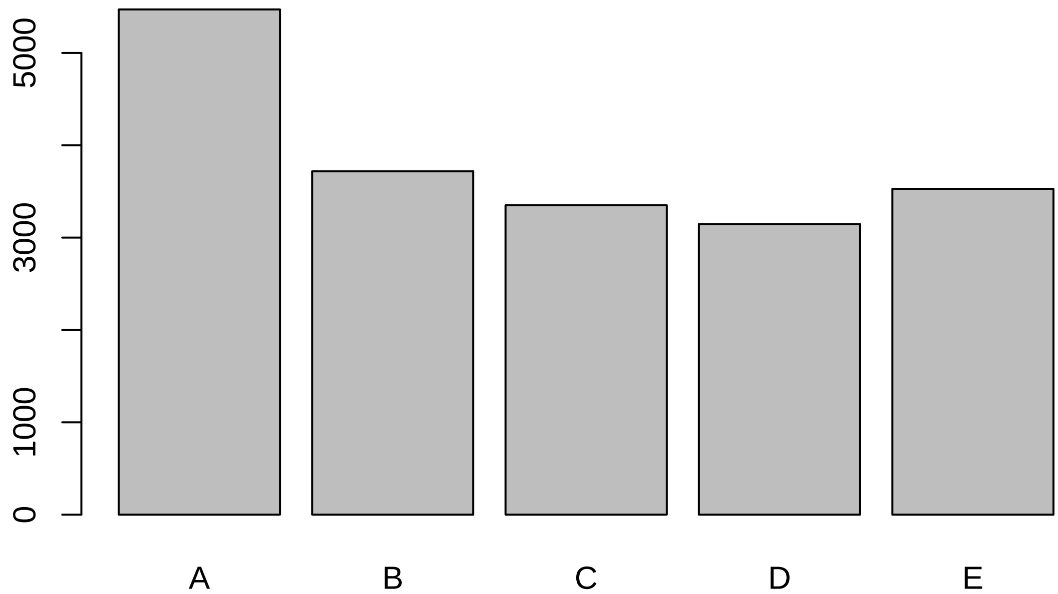
```r
ncol[nvar,]
```

```
##   [1] X                         user_name
##   [3] raw_timestamp_part_1      raw_timestamp_part_2
##   [5] cvtd_timestamp            new_window
##   [7] num_window                kurtosis_roll_belt
##   [9] kurtosis_picth_belt       kurtosis_yaw_belt
##  [11] kurtosis_roll_arm         kurtosis_picth_arm
##  [13] kurtosis_yaw_arm          kurtosis_roll_dumbbell
##  [15] kurtosis_picth_dumbbell   kurtosis_yaw_dumbbell
##  [17] kurtosis_roll_forearm     kurtosis_picth_forearm
##  [19] kurtosis_yaw_forearm      max_roll_belt
##  [21] max_picth_belt            max_yaw_belt
##  [23] max_roll_arm              max_picth_arm
##  [25] max_yaw_arm               max_roll_dumbbell
##  [27] max_picth_dumbbell        max_yaw_dumbbell
##  [29] max_roll_forearm          max_picth_forearm
##  [31] max_yaw_forearm           min_roll_belt
##  [33] min_pitch_belt            min_yaw_belt
##  [35] min_roll_arm              min_pitch_arm
##  [37] min_yaw_arm               min_roll_dumbbell
```

```
##  [39] min_pitch_dumbbell       min_yaw_dumbbell
##  [41] min_roll_forearm         min_pitch_forearm
##  [43] min_yaw_forearm          skewness_roll_belt
##  [45] skewness_roll_belt.1     skewness_yaw_belt
##  [47] skewness_roll_arm        skewness_pitch_arm
##  [49] skewness_yaw_arm         skewness_roll_dumbbell
##  [51] skewness_pitch_dumbbell  skewness_yaw_dumbbell
##  [53] skewness_roll_forearm    skewness_pitch_forearm
##  [55] skewness_yaw_forearm     amplitude_roll_belt
##  [57] amplitude_pitch_belt     amplitude_yaw_belt
##  [59] amplitude_roll_arm       amplitude_pitch_arm
##  [61] amplitude_yaw_arm        amplitude_roll_dumbbell
##  [63] amplitude_pitch_dumbbell amplitude_yaw_dumbbell
##  [65] amplitude_roll_forearm   amplitude_pitch_forearm
##  [67] amplitude_yaw_forearm    var_total_accel_belt
##  [69] var_roll_belt            var_pitch_belt
##  [71] var_yaw_belt             var_accel_arm
##  [73] var_roll_arm             var_pitch_arm
##  [75] var_yaw_arm              var_accel_dumbbell
##  [77] var_roll_dumbbell        var_pitch_dumbbell
##  [79] var_yaw_dumbbell         var_accel_forearm
##  [81] var_roll_forearm         var_pitch_forearm
##  [83] var_yaw_forearm          avg_roll_belt
##  [85] avg_pitch_belt           avg_yaw_belt
##  [87] avg_roll_arm             avg_pitch_arm
##  [89] avg_yaw_arm              avg_roll_dumbbell
##  [91] avg_pitch_dumbbell       avg_yaw_dumbbell
##  [93] avg_roll_forearm         avg_pitch_forearm
##  [95] avg_yaw_forearm          stddev_roll_belt
##  [97] stddev_pitch_belt        stddev_yaw_belt
##  [99] stddev_roll_arm          stddev_pitch_arm
## [101] stddev_yaw_arm           stddev_roll_dumbbell
## [103] stddev_pitch_dumbbell    stddev_yaw_dumbbell
## [105] stddev_roll_forearm      stddev_pitch_forearm
## [107] stddev_yaw_forearm
## 160 Levels: accel_arm_x accel_arm_y accel_arm_z ... yaw_forearm
```

The following graph shows the distribution of the response variable.

```r
plot(traincv$classe)
```



Data Partition

Train `traincv` variable contain only 53 variables including the objective variable. We limit our exploratory analisys to dimentionality reduction. The next step is data partition. Wi will partition the data in 70-30 proportion.

```
library(caret)
set.seed(123)

inTrain <- createDataPartition(y=traincv$classe, p=0.7, list=FALSE)

train <- traincv[inTrain, ]
test  <- traincv[-inTrain,]
```

Regression Tree Model

The model consists in a decission tree using `rpart` function that considers `classe` as dependent variable.

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

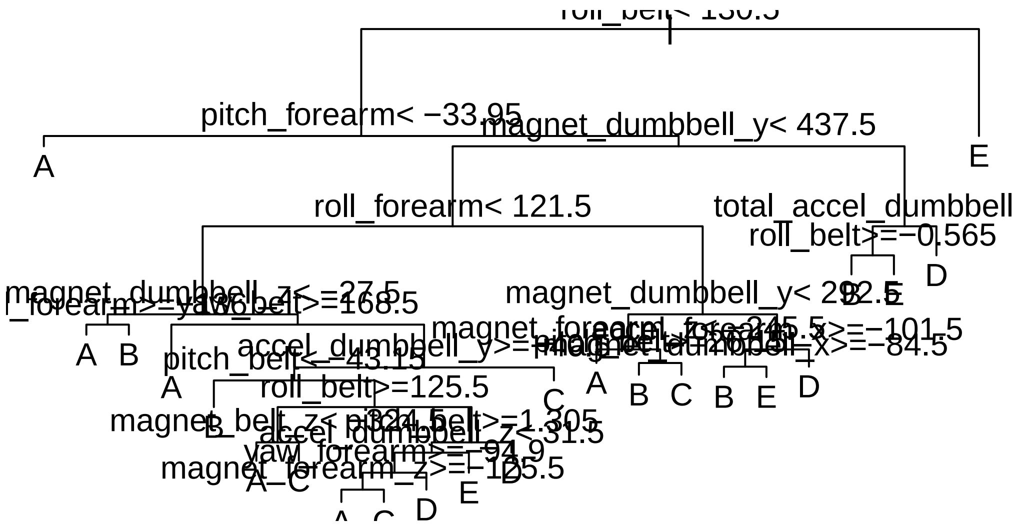
```
## Rattle: A free graphical interface for data mining with R.
```

```
## Versión 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Escriba 'rattle()' para agitar, sacudir y  rotar sus datos.
```

```r
fit1 <- rpart(classe ~ ., data=train, method="class")
```

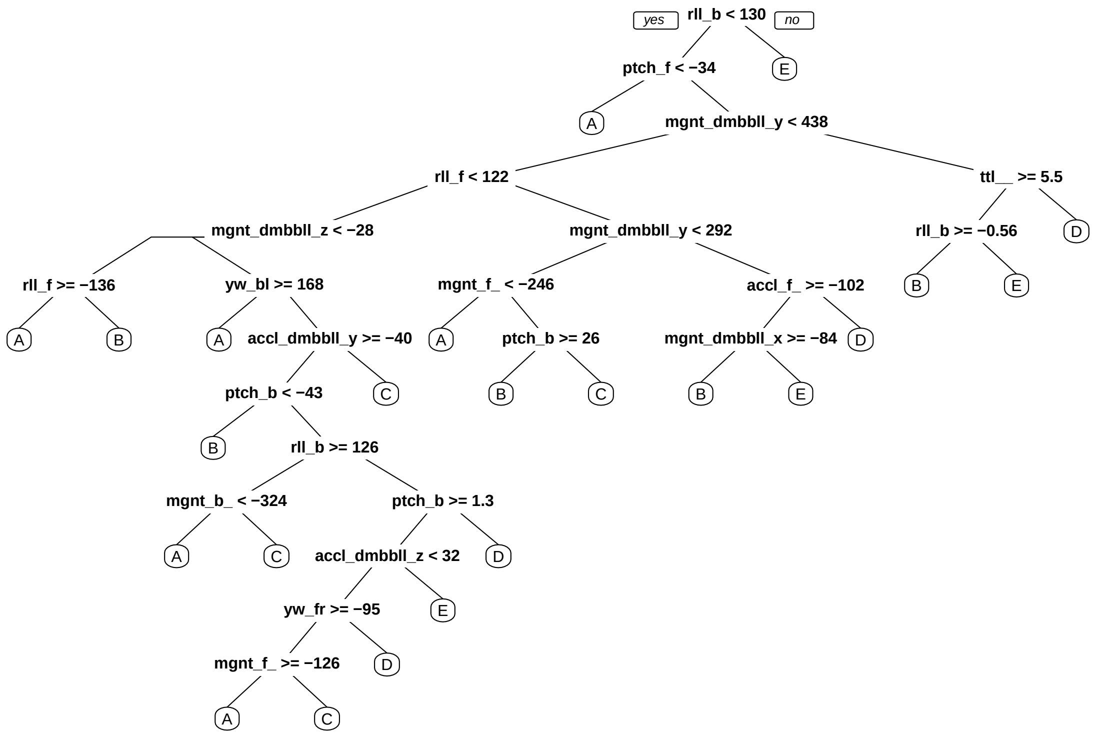
A first view of the tree.

```r
plot(fit1)
text(fit1)
```



A better view.

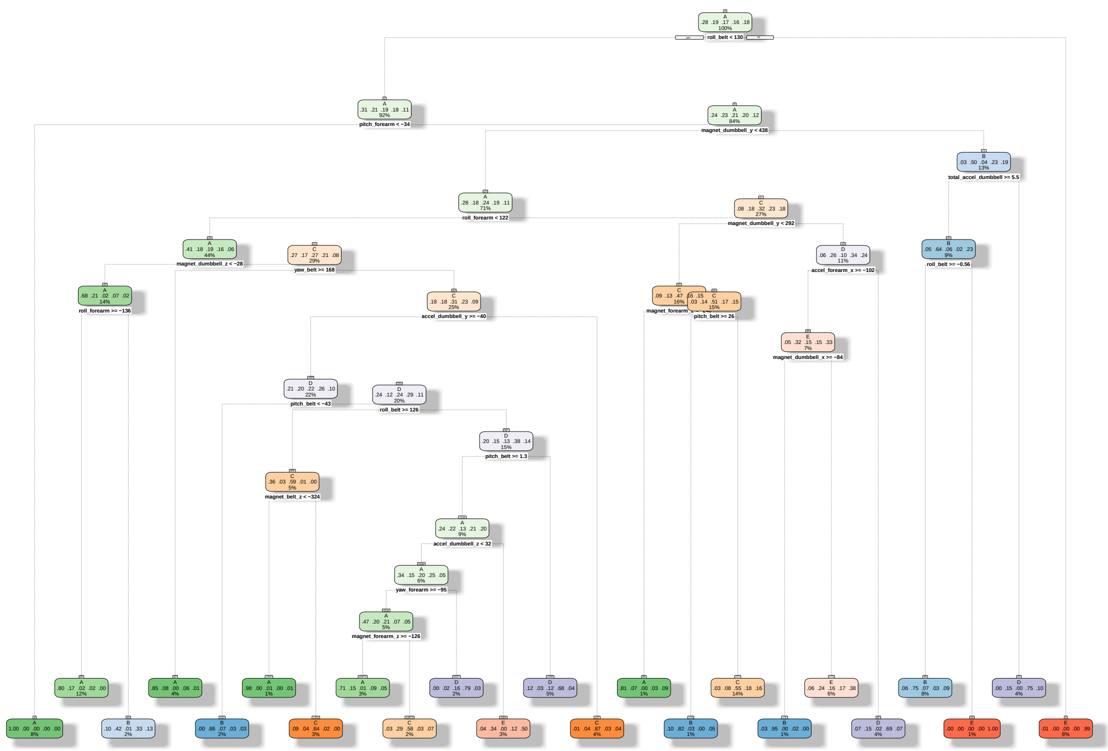
```r
prp(fit1,varlen=5)
```



There is another view using `rattle` package.

```
fancyRpartPlot(fit1)
```

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



Rattle 2016−ago.−14 22:35:59 Fernando

Using this model, we obtain the prediction of the 30% of the observations and build a confussion matrix

```
prediction <- predict(fit1, test, type="class")
confusionMatrix(prediction, test$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1450  166   11   58   22
##          B   58  612   44   64   65
##          C   42  124  798  126  115
##          D   62   74   80  624   57
##          E   29  139   72   72  799
##
## Overall Statistics
##
##                Accuracy : 0.7432
##                  95% CI : (0.7317, 0.7544)
##     No Information Rate : 0.2847
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6749
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
```

```
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8836   0.5489   0.7940   0.6610   0.7552
## Specificity            0.9377   0.9503   0.9145   0.9433   0.9337
## Pos Pred Value         0.8494   0.7260   0.6622   0.6957   0.7192
## Neg Pred Value         0.9529   0.8978   0.9546   0.9342   0.9443
## Prevalence             0.2847   0.1935   0.1744   0.1638   0.1836
## Detection Rate         0.2516   0.1062   0.1385   0.1083   0.1386
## Detection Prevalence   0.2962   0.1463   0.2091   0.1556   0.1928
## Balanced Accuracy      0.9106   0.7496   0.8542   0.8022   0.8444
```

Random forest Model

We build a second model.

```
library(randomForest)
fit2 <- randomForest(train$classe ~ .,   data=train, do.trace=F)
print(fit2)
```

```
##
## Call:
##  randomForest(formula = train$classe ~ ., data = train, do.trace = F)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.43%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3828    2    0    0    0 0.0005221932
## B    9 2588    6    0    0 0.0057625816
## C    0    9 2334    4    0 0.0055389859
## D    0    0   22 2179    2 0.0108942351
## E    0    0    3    1 2466 0.0016194332
```

Acording with results, this model is much more better than the decission tree, with only an error rate of 0.43%.

Prediction

Finally, beacuse of the low error rate, we choose random forest model and predict new observations.

```
new_observations_predict <- predict(fit2, testcv, type="class")
new_observations_predict
```

```
##  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
```

This example sohows the adventage in predictions of the random forest tree versus a simgle decission tree, but an the cost of high resources.