Prediction Models of a Weight Lifting Exercise

Endless-Void

7/26/2020

Introduction.

Devices like "Jawbone Up", "Nike FuelBand", and "Fitbit" allow us to to quantitatively analyze the movement of the people who wears it, this can be use to learn about the mistakes maded in certains exercises, in this case we will analyse and contruct a prediction model of the performance of a weight lifting exercise.

Analysis process.

Used libraries.

For the base code we use the next packages.

```
library(ggplot2)
library(caret)
library(dplyr)
library(ggpubr)
```

Data Obtention and Reading.

```
if(!file.exists("./data")){dir.create("./data")}
GeneralFiles <- list.files("./data", full.names = TRUE)
fileUrl1 <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
fileUrl2 <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
if(!file.exists("./data/training.csv")){
        download.file(fileUrl1, destfile = "./data/training.csv", method = "curl")
}
if(!file.exists("./data/testing.csv")){
        download.file(fileUrl2, destfile = "./data/testing.csv", method = "curl")
}
training <- read.csv("./data/training.csv", header = TRUE)
testing <- read.csv("./data/testing.csv", header = TRUE)
```

Pre-processing Data Sets.

A very easy way to see initial problems in the data sets its the "str" function.

```
str(training, list.len = 20)
```

```
## 'data.frame':    19622 obs. of  160 variables:
##  $ X                       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name               : chr  "carlitos" "carlitos" "carlitos" "carlitos" ...
##  $ raw_timestamp_part_1    : int  1323084231 1323084231 1323084231 1323084232 1323084232 1323084232 1
##  $ raw_timestamp_part_2    : int  788290 808298 820366 120339 196328 304277 368296 440390 484323 484
##  $ cvtd_timestamp          : chr  "05/12/2011 11:23" "05/12/2011 11:23" "05/12/2011 11:23" "05/12/20
```

```
##  $ new_window              : chr  "no" "no" "no" "no" ...
##  $ num_window              : int  11 11 11 12 12 12 12 12 12 12 ...
##  $ roll_belt               : num  1.41 1.41 1.42 1.48 1.48 1.45 1.42 1.42 1.43 1.45 ...
##  $ pitch_belt              : num  8.07 8.07 8.07 8.05 8.07 8.06 8.09 8.13 8.16 8.17 ...
##  $ yaw_belt                : num  -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 -94.4 ...
##  $ total_accel_belt        : int  3 3 3 3 3 3 3 3 3 3 ...
##  $ kurtosis_roll_belt      : chr  "" "" "" "" ...
##  $ kurtosis_picth_belt     : chr  "" "" "" "" ...
##  $ kurtosis_yaw_belt       : chr  "" "" "" "" ...
##  $ skewness_roll_belt      : chr  "" "" "" "" ...
##  $ skewness_roll_belt.1    : chr  "" "" "" "" ...
##  $ skewness_yaw_belt       : chr  "" "" "" "" ...
##  $ 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            : chr  "" "" "" "" ...
##   [list output truncated]
```

Note that we have a lot of variables with empty info and with NA values also some variables that dosent have relevant use for the machine learning part (the first 7 columns), so wee have to get rid of all of them.

First the NA Values.

```
NAbyCol <- apply(training, 2, function(x) sum(is.na(x)))
NACol <- names(subset(NAbyCol, NAbyCol == 19216))
training <- select(training, !NACol)
dim(training)
```

```
## [1] 19622    93
```

Then the empty and unnecessary columns.

```
training <- select(training, !names(training)[1:7])
vars2 <- grepl("kurtosis|skewness|amplitude|min|max", names(training))
training <- select(training, c(names(training[,!vars2]), classe))
dim(training)
```

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

Cross Validation.

```
set.seed(2549)
inTrain <- createDataPartition(y=training$classe,
                               p = 0.75, list = FALSE)
training <- training[inTrain,]
TEST <- training[-inTrain,]
dim(training)
```

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

```
dim(TEST)
```

```
## [1] 3651   53
```

Exploration Analysis.

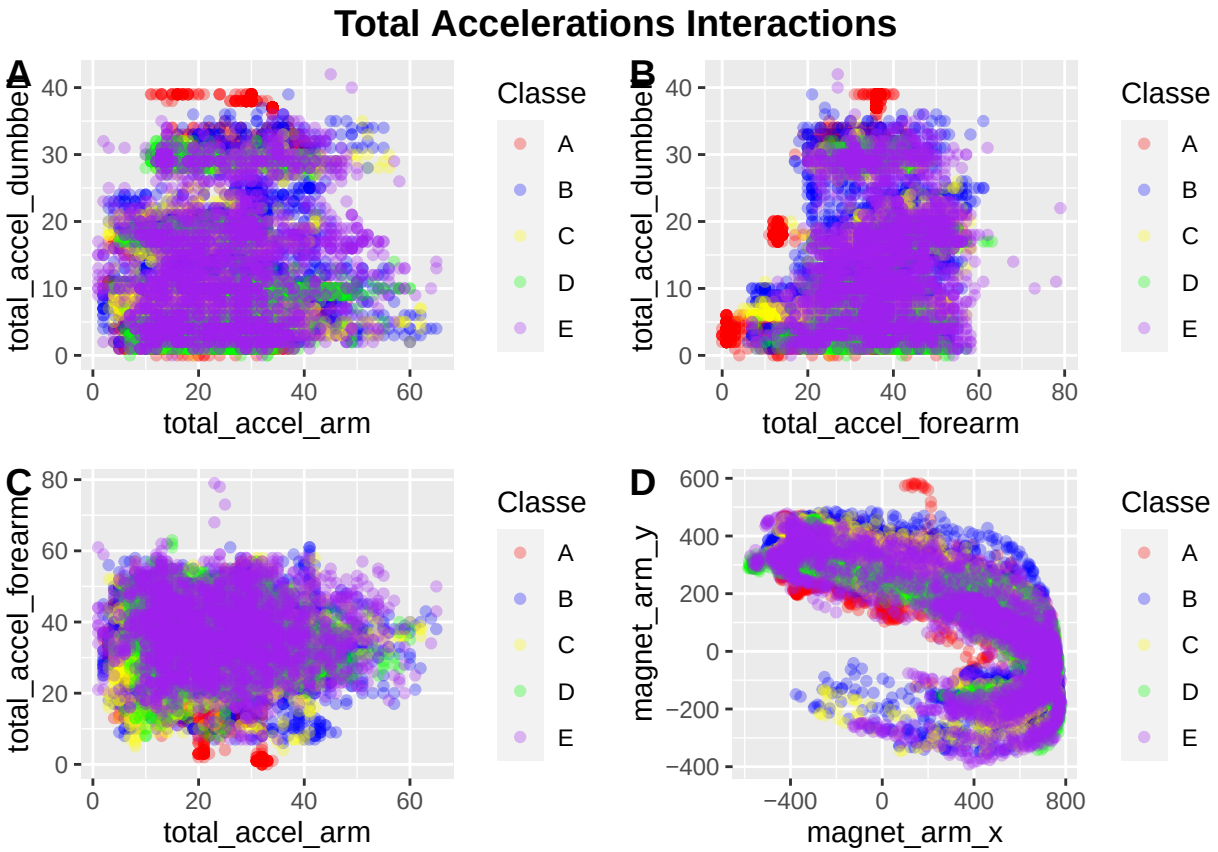

Figure 1.A to 1.C => Show the bidimensional interaction between the total acceleration recorded by the sensor of the arm, forearm and dumbell

Figure 1.D => Its an representative example of the bidimensional (X-Y plane) behavior of one of the sensors, in this case its from the arm

We can see two major things with this figure:

- A linear model can not explain this behavior.
- A Cluster analysis it's a waste of time.

Training Step.

For computational impairments, we will restric the internal iterations of some methods to avoid memory problems. we will use 2 iterations and a simple cross validation method for the internal resampling method in each strategy of training.

```
IteControl <- trainControl(method='cv', number = 2)
```

In order we will do:

1. Trees.
2. Trees with a pre-processing step using cluster analysis by the k-means method.
3. Random Forest.
4. Boosting with the tree method.

```
fit1 <- train(classe ~ ., method = "rpart", data = training)
fit1_1 <- train(classe ~ ., method = "rpart", preProcess = "knnImpute",data = training)
```

```
fit2 <- train(classe ~ ., method = "rf", trControl=IteControl, data = training, verbose=FALSE)
fit3 <- train(classe ~ ., method = "treebag", trControl=IteControl, data = training, verbose=FALSE)
```

For each one we have a initial Accuracy of:

1. 51.52%
2. 50.72% **Here we can see that a cluster analysis indeed it is a waste of time**
3. 98.56%
4. 97.11%

Selecting the best Method.

using our testing data set we get:

1. Trees.

```
confusionMatrix(as.factor(TEST$classe),predict(fit1, TEST))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 918  21  81   0   3
##          B 293 273 173   0   0
##          C 290  29 321   0   0
##          D 243 103 242   0   0
##          E  86 101 188   0 286
##
## Overall Statistics
##
##                Accuracy : 0.4925
##                  95% CI : (0.4761, 0.5088)
##     No Information Rate : 0.5012
##     P-Value [Acc > NIR] : 0.859
##
##                   Kappa : 0.3389
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.5016  0.51803  0.31940       NA  0.98962
## Specificity            0.9423  0.85083  0.87944   0.8389  0.88846
## Pos Pred Value         0.8974  0.36942  0.50156       NA  0.43268
## Neg Pred Value         0.6530  0.91277  0.77283       NA  0.99900
## Prevalence             0.5012  0.14434  0.27527   0.0000  0.07916
## Detection Rate         0.2514  0.07477  0.08792   0.0000  0.07833
## Detection Prevalence   0.2802  0.20241  0.17529   0.1611  0.18105
## Balanced Accuracy      0.7220  0.68443  0.59942       NA  0.93904
```

2. Trees with a pre-processing step using cluster analysis by the k-means method.

```
confusionMatrix(as.factor(TEST$classe),predict(fit1_1, TEST))
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction   A   B   C   D   E
##          A 918  21  81   0   3
##          B 293 273 173   0   0
##          C 290  29 321   0   0
##          D 243 103 242   0   0
##          E  86 101 188   0 286
##
## Overall Statistics
##
##                Accuracy : 0.4925
##                  95% CI : (0.4761, 0.5088)
##     No Information Rate : 0.5012
##     P-Value [Acc > NIR] : 0.859
##
##                   Kappa : 0.3389
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.5016  0.51803  0.31940       NA  0.98962
## Specificity            0.9423  0.85083  0.87944   0.8389  0.88846
## Pos Pred Value         0.8974  0.36942  0.50156       NA  0.43268
## Neg Pred Value         0.6530  0.91277  0.77283       NA  0.99900
## Prevalence             0.5012  0.14434  0.27527   0.0000  0.07916
## Detection Rate         0.2514  0.07477  0.08792   0.0000  0.07833
## Detection Prevalence   0.2802  0.20241  0.17529   0.1611  0.18105
## Balanced Accuracy      0.7220  0.68443  0.59942       NA  0.93904
```

3. Random Forest.

```
confusionMatrix(as.factor(TEST$classe),predict(fit2, TEST))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1023    0    0    0    0
##          B    0  739    0    0    0
##          C    0    0  640    0    0
##          D    0    0    0  588    0
##          E    0    0    0    0  661
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.999, 1)
##     No Information Rate : 0.2802
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 1
##
##  Mcnemar's Test P-Value : NA
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   1.0000   1.0000    1.000
## Specificity            1.0000   1.0000   1.0000   1.0000    1.000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000    1.000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000    1.000
## Prevalence             0.2802   0.2024   0.1753   0.1611    0.181
## Detection Rate         0.2802   0.2024   0.1753   0.1611    0.181
## Detection Prevalence   0.2802   0.2024   0.1753   0.1611    0.181
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000    1.000
```

4. Boosting with the tree method.

```
confusionMatrix(as.factor(TEST$classe),predict(fit3, TEST))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1023    0    0    0    0
##          B    0  739    0    0    0
##          C    0    0  640    0    0
##          D    0    0    0  588    0
##          E    0    0    1    0  660
##
## Overall Statistics
##
##                Accuracy : 0.9997
##                  95% CI : (0.9985, 1)
##     No Information Rate : 0.2802
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9997
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   0.9984   1.0000   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   0.9997
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   0.9985
## Neg Pred Value         1.0000   1.0000   0.9997   1.0000   1.0000
## Prevalence             0.2802   0.2024   0.1756   0.1611   0.1808
## Detection Rate         0.2802   0.2024   0.1753   0.1611   0.1808
## Detection Prevalence   0.2802   0.2024   0.1753   0.1611   0.1810
## Balanced Accuracy      1.0000   1.0000   0.9992   1.0000   0.9998
```

We can se that the first two models have problems to predict the Class D, and the best one is the **Random Forest Method**

Model Predictions.

Now we will use formally our Random Forest method to predict the class of 20 Weight lifting Exercises.

```
pred1 <- predict(fit1, testing)
pred1_1 <- predict(fit1_1, testing)
pred2 <- predict(fit2, testing)
pred3 <- predict(fit3, testing)
data.frame(Trees = pred1, Trees_kmeans = pred1_1, Random_Forest = pred2, Boosting_Trees = pred3)
```

```
##    Trees Trees_kmeans Random_Forest Boosting_Trees
## 1      C            C             B              B
## 2      A            A             A              A
## 3      C            C             B              B
## 4      A            A             A              A
## 5      A            A             A              A
## 6      C            C             E              E
## 7      C            C             D              D
## 8      A            A             B              B
## 9      A            A             A              A
## 10     A            A             A              A
## 11     C            C             B              B
## 12     C            C             C              C
## 13     C            C             B              B
## 14     A            A             A              A
## 15     C            C             E              E
## 16     A            A             E              E
## 17     A            A             A              A
## 18     A            A             B              B
## 19     A            A             B              B
## 20     C            C             B              B
```

```
pred2 ## Most accurate answer
```

```
##  [1] B A B A A E D B A A B C B A E E A B B B
## Levels: A B C D E
```