Excercise - Wearables

Luis F Botero

October 23, 2015

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.

```
library(caret)
```

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

```
library(doParallel)
```

```
## Loading required package: foreach
## Loading required package: iterators
## Loading required package: parallel
```

```
library(rattle)
```

```
## Loading required package: RGtk2
## Rattle: A free graphical interface for data mining with R.
## Version 3.5.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(RGtk2)
library(randomForest)
```

```
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
```

```
library(ggplot2)

#setwd(".\\Data_science\\Machine Learning\\Project")
## Original Training Set
training<-read.csv("pml-training.csv", header = TRUE,na.strings=c("DIV/0!"))
testing<-read.csv("pml-testing.csv", header = TRUE,na.strings=c("DIV/0!"))

# To ensure repetibility
set.seed(100)
```

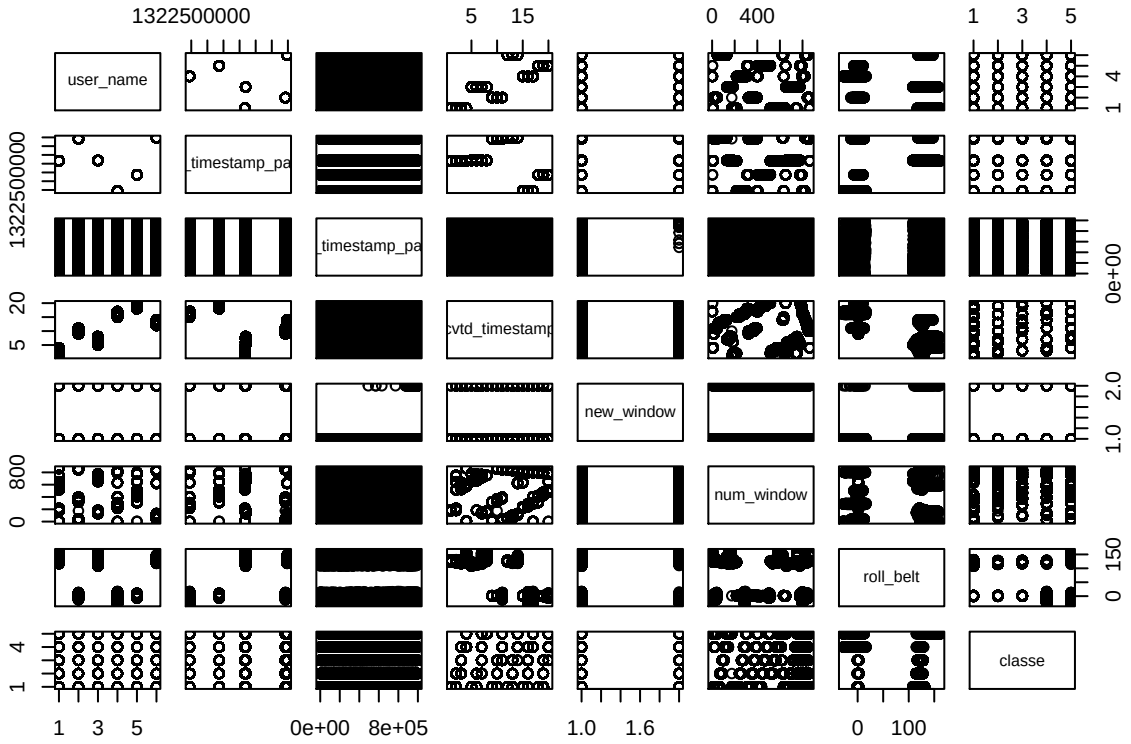
Exploratory Analysis

Just to get an idea, a few simple plots showing relationships between variables

```
str(training[,1:20])
```

```
## 'data.frame':    19622 obs. of  20 variables:
##  $ X                    : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ user_name            : Factor w/ 6 levels "adelmo","carlitos",..: 2 2 2 2 2 2 2 2 2 2 ...
##  $ raw_timestamp_part_1: int  1323084231 1323084231 1323084231 1323084232 1323084232 1323084232 1323
##  $ 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        : Factor w/ 196 levels "-0.2","-0.4",..: 196 196 196 196 196 196 196 196 196 1
##  $ max_picth_belt       : Factor w/ 23 levels "10","11","17",..: 23 23 23 23 23 23 23 23 23 23 ...
##  $ max_yaw_belt         : Factor w/ 68 levels "","-0.1","-0.2",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
selected_columns<-c(2:8,160)
pairs(training[,selected_columns])
```



From this pairs plot we can select some variables that are not randomly distributed, can be taken out of the model: timestamp, new windows, user name

Split Training and Validation sets

```
inTrain = createDataPartition(training$classe, p = .6)[[1]]
training = training[ inTrain,]
validation = training[-inTrain,]
```

Preprocessing

```
## Simplified discarding Near Zero Variability vars
NSV<-nearZeroVar(training)
training <- training[,-NSV]

# Eliminate first columns that are irrelevant to the prediction: timestamp,new windows,.
IR_COL<-grep("^user|timestamp|window",names(training))
training<-training[,-IR_COL]
```

Then we check for correlated columns and take them out to simplify the model

```
#Correlated variables
correlated_columns<-findCorrelation(cor(training[,-ncol(training)]),cutoff=.9,names=FALSE,exact=TRUE)
colnames(training)[correlated_columns]
```

```
## [1] "accel_belt_z"     "roll_belt"        "accel_belt_y"
```

```
## [4] "accel_belt_x"     "gyros_dumbbell_x" "gyros_dumbbell_z"
## [7] "gyros_arm_x"
```

```
training<-training[,-correlated_columns]
```

Model creation

```
# Use parallel to speed up calculations
registerDoParallel()
starting_time<-Sys.time()
model<-train(classe~.,data=training,method="rf",preProcess="pca",tuneGrid=data.frame(mtry=2))
ending_time<-Sys.time()
```

Model's calculation time

```
# I added it since it was taking so long without the Parallel, wanted to compare
print(ending_time-starting_time)
```

```
## Time difference of 5.371014 mins
```

The Random Forest method enables to calculate error and accuracy without a testing or cross validation set. Below the Accuracy and calculated error rate

Accuracy

```
print(model$results)
```

```
##   mtry  Accuracy     Kappa  AccuracySD     KappaSD
## 1    2 0.9798164 0.9744578 0.002700248 0.003403503
```

Error Rate

```
print(model$finalModel)
```

```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 1.21%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3344    4    0    0    0 0.001194743
## B   23 2234   21    1    0 0.019745502
## C    1   24 2020    9    0 0.016553067
## D    0    1   40 1884    5 0.023834197
## E    0    0    4   10 2151 0.006466513
```

Prediction out of training matriz

```
predict_training<-predict(model,training)
confusionMatrix(predict_training,training$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 3348    0    0    0    0
##          B    0 2279    0    0    0
##          C    0    0 2054    0    0
##          D    0    0    0 1930    0
##          E    0    0    0    0 2165
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9997, 1)
##     No Information Rate : 0.2843
##     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   1.0000
## Specificity            1.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000   1.0000   1.0000   1.0000   1.0000
## Prevalence             0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1935   0.1744   0.1639   0.1838
## Detection Prevalence   0.2843   0.1935   0.1744   0.1639   0.1838
## Balanced Accuracy      1.0000   1.0000   1.0000   1.0000   1.0000
```

Cross Validation

```
validation<-validation[,-NSV]
validation<-validation[,-IR_COL]
validation<-validation[,-correlated_columns]
predict_validation <-predict(model,newdata=validation)
confusionMatrix(predict_validation,validation$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1327    0    0    0    0
##          B    0  922    0    0    0
##          C    0    0  821    0    0
##          D    0    0    0  766    0
##          E    0    0    0    0  868
```

```
##
## Overall Statistics
##
##                Accuracy : 1
##                  95% CI : (0.9992, 1)
##     No Information Rate : 0.2821
##     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    1.000   1.0000   1.0000   1.0000
## Specificity            1.0000    1.000   1.0000   1.0000   1.0000
## Pos Pred Value         1.0000    1.000   1.0000   1.0000   1.0000
## Neg Pred Value         1.0000    1.000   1.0000   1.0000   1.0000
## Prevalence             0.2821    0.196   0.1745   0.1628   0.1845
## Detection Rate         0.2821    0.196   0.1745   0.1628   0.1845
## Detection Prevalence   0.2821    0.196   0.1745   0.1628   0.1845
## Balanced Accuracy      1.0000    1.000   1.0000   1.0000   1.0000
```

Now using the testing vector write the testing files

```
predict_testing<-predict(model,newdata=testing)
predict_testing
```

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

```
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)
  }
}
# Write files for the course testing
pml_write_files( as.character(predict_testing))
```