# PML Projet

***PML Student - MLO***

***Saturday, June 20, 2015***

# Prediction of Exercises Method from Accelerometer Data

## Introduction

This problem focuses on using a data set that has both incorrect and correct execution of several exercises. Motion was measured by accelerometers attached to the body. belt, forearm, arm, and dumbbell of 6 exercisers. They performed barbell lifts correctly and incorrectly in five different ways. The challenge is to build a classifier that identifies the exercise form (one of five) used by each exerciser. The data comes from http://groupware.les.inf.puc-rio.br/har (http://groupware.les.inf.puc-rio.br/har).

## Program and Hardware

The program selected to do the classification was based on the random forest classifier “rf” operating on the specified accelerometer data from the data set. The training set was used to train the model using the caret package from R. Compute time was more than expected (~ 15 minutes) but when the model was applied to the test data set, all test cases were resolved correctly.
Hardware used: Intel i7 4790 cpu, Intel Z97 chipset, 32 GB RAM, nVidia Titan X graphics.

If you run the ‘hidden’ code, I have simply included a read.csv(“url”) which will place the pml-training and pml-testing files in your current working directory.

I did create a feature plot of the predictor pairs for aldemo (commented out in the embedded code as it to several minutes of compute time) which was a colorful 12x12 array of dot plots that showed the processing a classifier would have to do. Interesting that train(…, method=“rf”) worked in the end for all of the test cases. I may become a believer.

```
##
## 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: 5.81%
## Confusion matrix:
##      A   B   C   D   E class.error
## A 1143   8   8   5   1  0.01888412
## B   18 719  28   8   3  0.07345361
## C   13  27 702   7   1  0.06400000
## D    6   4  66 432   7  0.16116505
## E    3   6   0   7 670  0.02332362


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


##
## 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: 5.59%
## Confusion matrix:
##     A   B   C   D   E class.error
## A 791   5   6  31   1  0.05155875
## B  22 629  32   2   5  0.08840580
## C   4   7 473   9   0  0.04056795
## D  19   5  14 448   0  0.07818930
## E   0   5   4   3 597  0.01970443


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

```
##
## 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: 3.28%
## Confusion matrix:
##     A   B   C   D   E class.error
## A 881  11   0   7   0  0.02002225
## B  30 695   9   8   3  0.06711409
## C   1   3 532   0   3  0.01298701
## D  10   0  13 613   6  0.04517134
## E   0   0   6   6 699  0.01687764


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


##
## 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: 1.66%
## Confusion matrix:
##     A   B   C   D   E class.error
## A 853   9   3   0   0 0.013872832
## B   6 578   7   0   1 0.023648649
## C   6   9 474   0   0 0.030674847
## D   0   0   3 573   6 0.015463918
## E   0   0   0   1 541 0.001845018


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

```
##
## 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: 5.73%
## Confusion matrix:
##      A   B   C   D   E class.error
## A 1119   6  39  12   1  0.04927782
## B   15 438  23   2  11  0.10429448
## C   13  16 615   6   2  0.05674847
## D    5   0  12 503   2  0.03639847
## E    0  12  13   5 532  0.05338078


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


##
## 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: 5.52%
## Confusion matrix:
##     A   B   C   D   E class.error
## A 604   4   8  24   0  0.05625000
## B  23 472   7   0   3  0.06534653
## C  17  13 467   2   0  0.06412826
## D  14   0  14 441   0  0.05970149
## E   2  11   1   1 482  0.03018109


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

# Cross-Validation

The training set was partitioned to create a cross-validation set of training/test data using the createDataPartition() function from test subject jeremy.

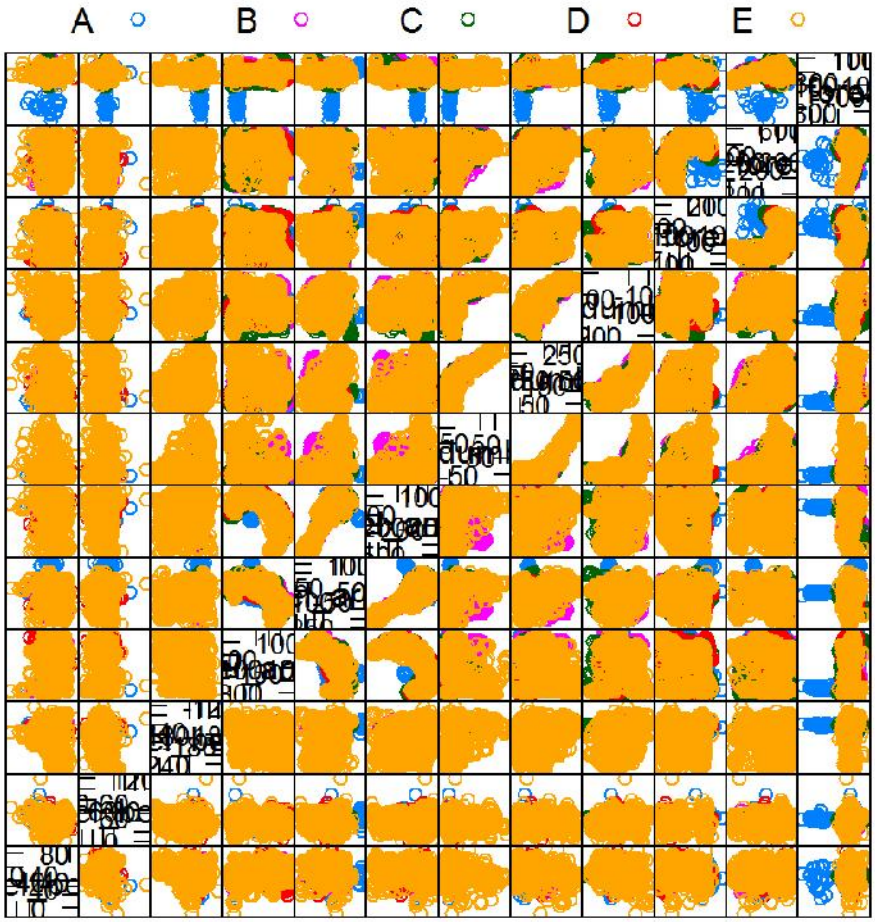
```
##
## Call:
##  randomForest(x = x, y = y, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 6.38%
## Confusion matrix:
##     A   B   C   D   E class.error
## A 835   8  30   9   1  0.05436014
## B  15 324  21   0   7  0.11716621
## C   9  11 459   8   2  0.06134969
## D   3   0  12 377   0  0.03826531
## E   1   8  14   4 395  0.06398104


##
## 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: 6.93%
## Confusion matrix:
##     A   B   C   D   E class.error
## A 830   6  35  11   1  0.06002265
## B  13 321  21   1  11  0.12534060
## C  11  15 457   5   1  0.06543967
## D   5   1  10 374   2  0.04591837
## E   0   8  15   5 394  0.06635071
```

```
##
## 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: 6.74%
## Confusion matrix:
##     A   B   C   D   E class.error
## A 826   6  37  13   1  0.06455266
## B  12 325  23   1   6  0.11444142
## C   9  10 461   8   1  0.05725971
## D   3   1   9 377   2  0.03826531
## E   0   9  14   7 392  0.07109005
```

Three passes were averaged to derive estimates of data error, ~ 6.7 %, (0.0695869 from finalModel$confusion and 0.0636042 from prediction versus truth).

What follows is a feature plot by pairs for training data set of adelmo. This visually shows the challenge faced by the classifier algorithm.




Scatter Plot Matrix

# Conclusions and Remarks

The good news: * The caret package with the random forest training method worked successfully to correctly match all 20 test cases. * The cross-validation suggested error rates in the six to seven per cent accuracy.

Challenges: * Computation time for the random forest method was excessive in my opinion even with an above average personal computer. I probably should have done a principle components analysis to simplify variables. * There are way too many ways (for the novice especially) to be misled. One example mean(x1, x2, x3) gives a different result from mean(c(x1, x2, x3)). Add the black box nature of these model-fitting algorithms as implemented in caret, and you need to do a lot of checking of intermediate steps to make sure the results are sensible. Hence, graphics, printout, and cross-validation are necessary to build confidence in your results. * I find official R documentation to be annoyingly obscure in most cases. Web searching for answers such as including setInternet2(TRUE) before trying to knit a markdown file with a read.csv("https://(https://)…") is absolutely essential to fill in the explanation and examples gaps. * Finally, suggestions for improvement, especially faster processing are appreciated. And yes, I could have used loops to reduce lines of code. Maybe next time when I have more confidence that what's going on under the hood is what I expect. You know, the black box munching mystery.