Machine Learning Project

Kumar Velugula

June 21, 2015

Executive Summary

The goal of the project is to predict the manner (how well they have done it) in which a group of 6 enthusiasts did the exercises based on the training/test data collected from accelormeters fitted to different parts of their bodies. Particularly "classe" variable in the training set will be used to predict with.

This document is a report describing how model is built, how cross validation is used, what the expected out of sample error is, and why certain choices are made in due course.

Based on the detailed analysis presented below, it is possible to accurately predict how well the enthusiast is performing exercise using the Random Forest algorithm.

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

Note: To load data and initialize the models/plots it may take upto 1-2 minutes

```
#load dependencies
#Please make sure listed below packages are installed before proceeding
library(caret)
```

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

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

```
## Rattle: A free graphical interface for data mining with R.
## Version 3.4.1 Copyright (c) 2006-2014 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

```
library(randomForest)
```

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

Getting and Cleaning data

```
downloadSourceFiles <- function(){
        ##check if both the required files exists in the current working directory,
        ##if any one of them is missing, re-download the zip file and unzip to download both the files
        if(file.exists("pml-training.csv") == FALSE
           | file.exists("pml-testing.csv") == FALSE ){
```

```
                #training data location
                trainingUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
                #The test data are available here:
                testUrl <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

                ## download file with name "pml-training.csv"
                download.file(trainingUrl,dest="pml-training.csv", method="curl", mode="wb")
                ## download file with name "pml-testing.csv"
                download.file(testUrl,dest="pml-testing.csv", method="curl", mode="wb")

                print ('Required files 1. pml-training.csv and 2. pml-testing.csv are downloaded.')

        }
        else {
                print ('Required files 1. pml-training.csv and 2. pml-testing.csv already exists and no
        }
}
downloadSourceFiles();
```

```
## [1] "Required files 1. pml-training.csv and 2. pml-testing.csv already exists and not re-downloading
```

```
#all NA like values including blanks are explicitly set to be interpreted as NA values
pmlTrainingData <- read.csv("pml-training.csv", na.strings=c("NA","","  "))
pmlTestingData <- read.csv("pml-testing.csv", na.strings=c("NA","","  "))

#From training data remove columns with all NA values explicitly so that it won't interfere with modeli
naPredicate <- apply(pmlTrainingData, 2, function(itemVal) sum(is.na(itemVal)));
pmlTrainigDataCleaned <- pmlTrainingData[, which(naPredicate == 0)];

#From trainingDataCleaned, remove identifier columns, timestamp etc., as they are not relevant to the m
pmlTrainigDataCleaned <- pmlTrainigDataCleaned[8:length(pmlTrainigDataCleaned)]


naPredicateTest <- apply(pmlTestingData, 2, function(itemVal) sum(is.na(itemVal)));
pmlTestDataCleaned <- pmlTestingData[, which(naPredicateTest == 0)];

#From pmlTestDataCleaned, remove identifier columns, timestamp etc., as they are not relevant to the mo
pmlTestDataCleaned <- pmlTestDataCleaned[8:length(pmlTestDataCleaned)]

#Towards reproducibilty setting the seed
set.seed(654321)
```

Exploratory Data Analysis

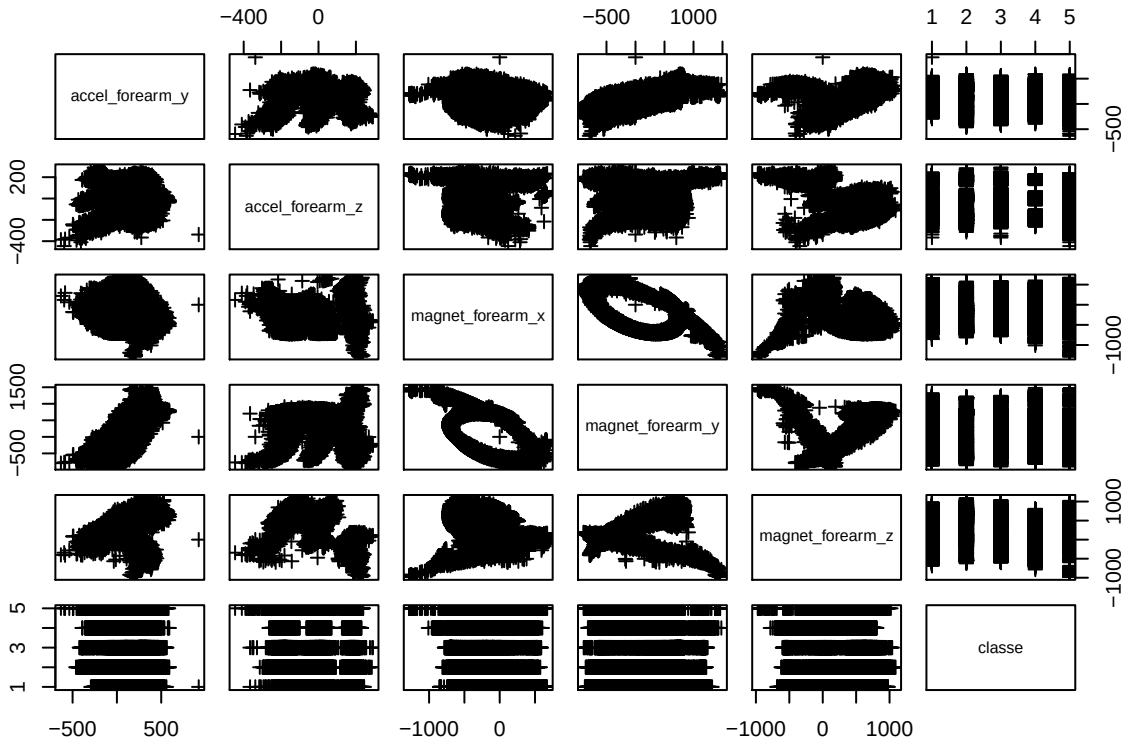
```
#Quick look at the cleaned data frame
str(pmlTrainigDataCleaned)
```

```
## 'data.frame':    19622 obs. of  53 variables:
##  $ 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 ...
##  $ gyros_belt_x        : num  0 0.02 0 0.02 0.02 0.02 0.02 0.02 0.02 0.03 ...
##  $ gyros_belt_y        : num  0 0 0 0 0.02 0 0 0 0 0 ...
##  $ gyros_belt_z        : num  -0.02 -0.02 -0.02 -0.03 -0.02 -0.02 -0.02 -0.02 -0.02 0 ...
##  $ accel_belt_x        : int  -21 -22 -20 -22 -21 -21 -22 -22 -20 -21 ...
##  $ accel_belt_y        : int  4 4 5 3 2 4 3 4 2 4 ...
##  $ accel_belt_z        : int  22 22 23 21 24 21 21 21 24 22 ...
##  $ magnet_belt_x       : int  -3 -7 -2 -6 -6 0 -4 -2 1 -3 ...
##  $ magnet_belt_y       : int  599 608 600 604 600 603 599 603 602 609 ...
##  $ magnet_belt_z       : int  -313 -311 -305 -310 -302 -312 -311 -313 -312 -308 ...
##  $ roll_arm            : num  -128 -128 -128 -128 -128 -128 -128 -128 -128 -128 ...
##  $ pitch_arm           : num  22.5 22.5 22.5 22.1 22.1 22 21.9 21.8 21.7 21.6 ...
##  $ yaw_arm             : num  -161 -161 -161 -161 -161 -161 -161 -161 -161 -161 ...
##  $ total_accel_arm     : int  34 34 34 34 34 34 34 34 34 34 ...
##  $ gyros_arm_x         : num  0 0.02 0.02 0.02 0 0.02 0 0.02 0.02 0.02 ...
##  $ gyros_arm_y         : num  0 -0.02 -0.02 -0.03 -0.03 -0.03 -0.03 -0.02 -0.03 -0.03 ...
##  $ gyros_arm_z         : num  -0.02 -0.02 -0.02 0.02 0 0 0 0 -0.02 -0.02 ...
##  $ accel_arm_x         : int  -288 -290 -289 -289 -289 -289 -289 -289 -288 -288 ...
##  $ accel_arm_y         : int  109 110 110 111 111 111 111 111 109 110 ...
##  $ accel_arm_z         : int  -123 -125 -126 -123 -123 -122 -125 -124 -122 -124 ...
##  $ magnet_arm_x        : int  -368 -369 -368 -372 -374 -369 -373 -372 -369 -376 ...
##  $ magnet_arm_y        : int  337 337 344 344 337 342 336 338 341 334 ...
##  $ magnet_arm_z        : int  516 513 513 512 506 513 509 510 518 516 ...
##  $ roll_dumbbell       : num  13.1 13.1 12.9 13.4 13.4 ...
##  $ pitch_dumbbell      : num  -70.5 -70.6 -70.3 -70.4 -70.4 ...
##  $ yaw_dumbbell        : num  -84.9 -84.7 -85.1 -84.9 -84.9 ...
##  $ total_accel_dumbbell: int  37 37 37 37 37 37 37 37 37 37 ...
##  $ gyros_dumbbell_x    : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ gyros_dumbbell_y    : num  -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 -0.02 ...
##  $ gyros_dumbbell_z    : num  0 0 0 -0.02 0 0 0 0 0 0 ...
##  $ accel_dumbbell_x    : int  -234 -233 -232 -232 -233 -234 -232 -234 -232 -235 ...
##  $ accel_dumbbell_y    : int  47 47 46 48 48 48 47 46 47 48 ...
##  $ accel_dumbbell_z    : int  -271 -269 -270 -269 -270 -269 -270 -272 -269 -270 ...
##  $ magnet_dumbbell_x   : int  -559 -555 -561 -552 -554 -558 -551 -555 -549 -558 ...
##  $ magnet_dumbbell_y   : int  293 296 298 303 292 294 295 300 292 291 ...
##  $ magnet_dumbbell_z   : num  -65 -64 -63 -60 -68 -66 -70 -74 -65 -69 ...
##  $ roll_forearm        : num  28.4 28.3 28.3 28.1 28 27.9 27.9 27.8 27.7 27.7 ...
##  $ pitch_forearm       : num  -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.9 -63.8 -63.8 -63.8 ...
##  $ yaw_forearm         : num  -153 -153 -152 -152 -152 -152 -152 -152 -152 -152 ...
##  $ total_accel_forearm : int  36 36 36 36 36 36 36 36 36 36 ...
##  $ gyros_forearm_x     : num  0.03 0.02 0.03 0.02 0.02 0.02 0.02 0.02 0.03 0.02 ...
##  $ gyros_forearm_y     : num  0 0 -0.02 -0.02 0 -0.02 0 -0.02 0 0 ...
##  $ gyros_forearm_z     : num  -0.02 -0.02 0 0 -0.02 -0.03 -0.02 0 -0.02 -0.02 ...
##  $ accel_forearm_x     : int  192 192 196 189 189 193 195 193 193 190 ...
##  $ accel_forearm_y     : int  203 203 204 206 206 203 205 205 204 205 ...
##  $ accel_forearm_z     : int  -215 -216 -213 -214 -214 -215 -215 -213 -214 -215 ...
##  $ magnet_forearm_x    : int  -17 -18 -18 -16 -17 -9 -18 -9 -16 -22 ...
##  $ magnet_forearm_y    : num  654 661 658 658 655 660 659 660 653 656 ...
##  $ magnet_forearm_z    : num  476 473 469 469 473 478 470 474 476 473 ...
##  $ classe              : Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
#Quick preview of the observed data in first 5 columns
#pairs(pmlTrainigDataCleaned[,c(1:5)], pch=3)
#Quick preview of the observed data in last 5 columns
```

```r
pairs(pmlTrainigDataCleaned[,c(48:53)], pch=3)
```



Create train/test partitions

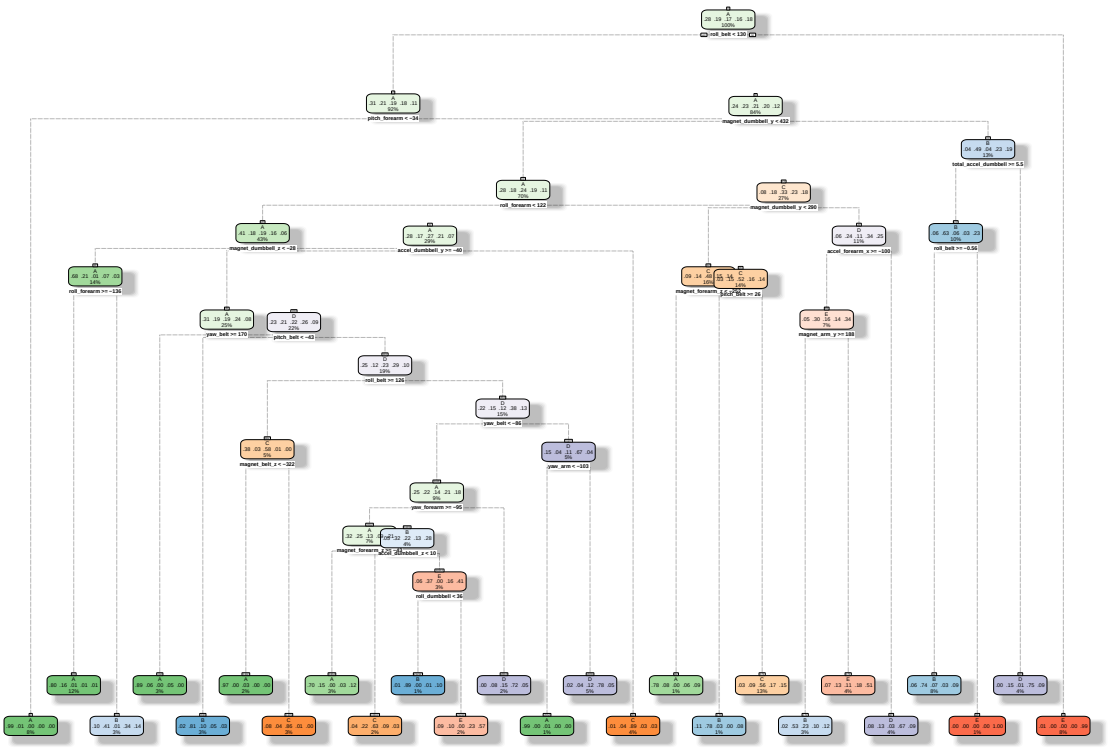
Splitting data set into 70/30 ratio for training and cross validation respectively

```r
#Split the pmlTrainigDataCleaned data into training and cross validation data sets
inTraining <- createDataPartition(y=pmlTrainigDataCleaned$classe, p=0.7, list=FALSE)
training <- pmlTrainigDataCleaned[inTraining,]
crossVal <- pmlTrainigDataCleaned[-inTraining,]
```

Using Decision Tree algorithm for prediction

```r
#generate the model using Decision Tree algorithm
dtModel <- rpart(classe ~., data=training, method="class")
#create model plot
fancyRpartPlot(dtModel)
```

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



Rattle 2015−Jun−21 16:05:42 gopala

```
#predict using the above model against the crossValidation set
dtPrediction <- predict(dtModel, crossVal,type="class")
#generate confusion matrix for reviewing the accuracy levels
confusionMatrix(dtPrediction, crossVal$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1512  153   13   32   48
##          B   66  736   91   76   90
##          C   49  109  828  169  135
##          D   13   81   64  609   59
##          E   34   60   30   78  750
##
## Overall Statistics
##
##                Accuracy : 0.7536
##                  95% CI : (0.7424, 0.7646)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6878
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
```

```
## Sensitivity            0.9032   0.6462   0.8070   0.6317   0.6932
## Specificity            0.9416   0.9319   0.9049   0.9559   0.9579
## Pos Pred Value         0.8601   0.6950   0.6419   0.7373   0.7878
## Neg Pred Value         0.9607   0.9165   0.9569   0.9298   0.9327
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2569   0.1251   0.1407   0.1035   0.1274
## Detection Prevalence   0.2987   0.1799   0.2192   0.1404   0.1618
## Balanced Accuracy      0.9224   0.7891   0.8560   0.7938   0.8256
```

As seen from the confusion matrix generated from the Decision tree prediction is only 76% and will be evaluating another modeling approach using Random Forest method for improved prediction accuracy

Using Random Forest algorithm for prediction

```
#generate the model using Random Forest algorithm
rfModel <- randomForest(classe ~ ., data=training)
#predict using the above model against the crossValidation set
rfPrediction <- predict(rfModel, crossVal, type="class")
#generate confusion matrix for reviewing the accuracy levels
confusionMatrix(rfPrediction, crossVal$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    0 1139    2    0    0
##          C    0    0 1024   11    5
##          D    0    0    0  952    4
##          E    0    0    0    1 1073
##
## Overall Statistics
##
##                Accuracy : 0.9961
##                  95% CI : (0.9941, 0.9975)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9951
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   1.0000   0.9981   0.9876   0.9917
## Specificity            1.0000   0.9996   0.9967   0.9992   0.9998
## Pos Pred Value         1.0000   0.9982   0.9846   0.9958   0.9991
## Neg Pred Value         1.0000   1.0000   0.9996   0.9976   0.9981
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1935   0.1740   0.1618   0.1823
## Detection Prevalence   0.2845   0.1939   0.1767   0.1624   0.1825
## Balanced Accuracy      1.0000   0.9998   0.9974   0.9934   0.9957
```

As seen from the confusion matrix generated from the RF prediction, it is 99.59% and got better results than that from Decision Tree algorithm. RF algorithm accuracy level is satisfactory for the current use case and proceeding to use the model to predict using the cleaned test data. The expected out of sample error rate is of the order of 0.41% based on the above accuracy levels from training/test data sets.

Use Random Forest model to predict the cleaned test data from the source

```
testPrediction <- predict(rfModel, pmlTestDataCleaned)
testPrediction
```

```
##  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
```

Generate files for each of the predictions

```
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(testPrediction);
```