Practical Machine Learning Project: Analysis to predict the manner in which exercise was done

melabraham

Sunday, January 18, 2015

Executive Summary

Given the test and training data from the source http://groupware.les.inf.puc-rio.br/har, the outcome of this project is to process the data gathered from accelerometers on the belt, forearm, arm, and dumbell of the participants in a machine learning algorithm and do a detailed analysis to predict the manner in which the participants did the exercise.

The prediction model is run on the test data to predict the outcome of 20 different test cases.

Reproducability

Step1 : Load the libraries

```
library(AppliedPredictiveModeling)
```

```
## Warning: package 'AppliedPredictiveModeling' was built under R version
## 3.1.2
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.1.2

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

## Warning: package 'ggplot2' was built under R version 3.1.1
```

```
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.1.2

## 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)
```

```
## Warning: package 'randomForest' was built under R version 3.1.2

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

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 3.1.2
```

Load and Clean data

Step2 : Import the data and verify that the training data and the test data are identical. Select the columns for analysis

```
# convert blank,NA and #DIV/0 to NA . Remove columns containing 'NA' from the downloaded datasets
df_training <- read.csv("pml-training.csv", na.strings=c("NA","","#DIV/0!"), header=TRUE)
colnames_train <- colnames(df_training)

df_testing <- read.csv("pml-training.csv", na.strings=c("NA","","#DIV/0!"), header=TRUE)
colnames_test <- colnames(df_testing)

## Eliminate other extraneous columns that are not needed

# Count the number of non-NAs in each col.
nonNAs <- function(x) {
    as.vector(apply(x, 2, function(x) length(which(!is.na(x)))))
}

# Build vector of missing data or NA columns to drop.
colcnts <- nonNAs(df_training)
drops <- c()
for (cnt in 1:length(colcnts)) {
    if (colcnts[cnt] < nrow(df_training)) {
        drops <- c(drops, colnames_train[cnt])
    }
}

# Drop NA data and the first 7 columns as they're unnecessary for predicting.
df_training <- df_training[,!(names(df_training) %in% drops)]
df_training <- df_training[,8:length(colnames(df_training))]

df_testing <- df_testing[,!(names(df_testing) %in% drops)]
df_testing <- df_testing[,8:length(colnames(df_testing))]

# Show remaining columns.
colnames(df_training)
```

```
##  [1] "roll_belt"             "pitch_belt"            "yaw_belt"
##  [4] "total_accel_belt"      "gyros_belt_x"          "gyros_belt_y"
##  [7] "gyros_belt_z"          "accel_belt_x"          "accel_belt_y"
## [10] "accel_belt_z"          "magnet_belt_x"         "magnet_belt_y"
## [13] "magnet_belt_z"         "roll_arm"              "pitch_arm"
## [16] "yaw_arm"               "total_accel_arm"       "gyros_arm_x"
## [19] "gyros_arm_y"           "gyros_arm_z"           "accel_arm_x"
## [22] "accel_arm_y"           "accel_arm_z"           "magnet_arm_x"
## [25] "magnet_arm_y"          "magnet_arm_z"          "roll_dumbbell"
## [28] "pitch_dumbbell"        "yaw_dumbbell"          "total_accel_dumbbell"
## [31] "gyros_dumbbell_x"     "gyros_dumbbell_y"     "gyros_dumbbell_z"
## [34] "accel_dumbbell_x"     "accel_dumbbell_y"     "accel_dumbbell_z"
## [37] "magnet_dumbbell_x"    "magnet_dumbbell_y"    "magnet_dumbbell_z"
## [40] "roll_forearm"         "pitch_forearm"        "yaw_forearm"
## [43] "total_accel_forearm"  "gyros_forearm_x"      "gyros_forearm_y"
## [46] "gyros_forearm_z"      "accel_forearm_x"      "accel_forearm_y"
## [49] "accel_forearm_z"      "magnet_forearm_x"     "magnet_forearm_y"
## [52] "magnet_forearm_z"     "classe"
```

```
colnames(df_testing)
```

```
##  [1] "roll_belt"            "pitch_belt"           "yaw_belt"
##  [4] "total_accel_belt"     "gyros_belt_x"         "gyros_belt_y"
##  [7] "gyros_belt_z"         "accel_belt_x"         "accel_belt_y"
## [10] "accel_belt_z"         "magnet_belt_x"        "magnet_belt_y"
## [13] "magnet_belt_z"        "roll_arm"             "pitch_arm"
## [16] "yaw_arm"              "total_accel_arm"      "gyros_arm_x"
## [19] "gyros_arm_y"          "gyros_arm_z"          "accel_arm_x"
## [22] "accel_arm_y"          "accel_arm_z"          "magnet_arm_x"
## [25] "magnet_arm_y"         "magnet_arm_z"         "roll_dumbbell"
## [28] "pitch_dumbbell"       "yaw_dumbbell"         "total_accel_dumbbell"
## [31] "gyros_dumbbell_x"     "gyros_dumbbell_y"     "gyros_dumbbell_z"
## [34] "accel_dumbbell_x"     "accel_dumbbell_y"     "accel_dumbbell_z"
## [37] "magnet_dumbbell_x"    "magnet_dumbbell_y"    "magnet_dumbbell_z"
## [40] "roll_forearm"         "pitch_forearm"        "yaw_forearm"
## [43] "total_accel_forearm"  "gyros_forearm_x"      "gyros_forearm_y"
## [46] "gyros_forearm_z"      "accel_forearm_x"      "accel_forearm_y"
## [49] "accel_forearm_z"      "magnet_forearm_x"     "magnet_forearm_y"
## [52] "magnet_forearm_z"     "classe"
```

Split training data set into 80/20 subsamples

```
# seed random # gen for subsetting
set.seed(625)

inTrain <- createDataPartition(y = df_training$classe, p = 0.8, list = F)
trainingSub <- df_training[inTrain, ]
testingSub <- df_training[-inTrain, ]

dim(trainingSub)
```

```
## [1] 15699    53
```

```
dim(testingSub)
```

```
## [1] 3923   53
```

Using ML algorithms for prediction : Random Forests

Random Forests were used as there are 52 input variables and they are well suited to handle a large number of inputs, especially when the interactions between variables are unknown. Also, it has a built in cross-validation component that gives an unbiased estimate of the forest???s out-of-sample (OOB) error rate

```
modFitA1 <- randomForest(classe ~. , data=trainingSub)

# predicting in-sample error
predict_test <- predict(modFitA1, testingSub, type = "class")

# using confusion matrix to test results
confusionMatrix(predict_test, testingSub$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1116    1    0    0    0
##          B    0  755    4    0    0
##          C    0    3  680    4    4
##          D    0    0    0  639    0
##          E    0    0    0    0  717
##
## Overall Statistics
##
##                Accuracy : 0.996
##                  95% CI : (0.993, 0.998)
##     No Information Rate : 0.284
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.995
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity             1.000    0.995    0.994    0.994    0.994
## Specificity             1.000    0.999    0.997    1.000    1.000
## Pos Pred Value          0.999    0.995    0.984    1.000    1.000
## Neg Pred Value          1.000    0.999    0.999    0.999    0.999
## Prevalence              0.284    0.193    0.174    0.164    0.184
## Detection Rate          0.284    0.192    0.173    0.163    0.183
## Detection Prevalence    0.285    0.193    0.176    0.163    0.183
## Balanced Accuracy       1.000    0.997    0.995    0.997    0.997
```

random forests yielded good results.

Out of sample error

The out of sample error after running the predict() function on the test set : 1 - 0.996 = 0.014

Generating files for submission

```
 # Using the provided Test set out-of-sample error
predictionsB2 <- as.character(predict(modFitA1, testingSub))


 # Function to generate files with prediction to submit for assignment

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(predictionsB2)
```