PML1

Girija Attigeri

January 10, 2019

Summary

This report uses machine learning algorithms to predict the manner in which users of exercise devices exercise.

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. More information is available from the website here: (see the section on the Weight Lifting Exercise Dataset).

Data

The training data for this project are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv

The test data are available here:

https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv

Set the work environment and knitr options

```
#setwd("C://Users//MAHE//Documents//PMLData//")
rm(list=ls(all=TRUE)) #start with empty workspace
startTime <- Sys.time()
library(knitr)
```

```
## Warning: package 'knitr' was built under R version 3.5.2
```

```
opts_chunk$set(echo = TRUE, cache= TRUE, results = 'hold')
```

Load libraries and Set Seed

Load all libraries used, and setting seed for reproducibility. *Results Hidden, Warnings FALSE and Messages FALSE*

```
library(ElemStatLearn)
library(caret)
library(rpart)
library(randomForest)
```

```
library(RCurl)
set.seed(2014)
```

Load and prepare the data and clean up the data

Load and prepare the data

```
pml_CSV<- read.csv("C://Users//MAHE//Documents//PMLData//pml-training.csv", header=TRUE, sep=",", na.st
pml_CSV <- pml_CSV[,-1] # Remove the first column that represents a ID Row
```

Data Sets Partitions Definitions

Create data partitions of training and validating data sets.

```
inTrain = createDataPartition(pml_CSV$classe, p=0.60, list=FALSE)
training = pml_CSV[inTrain,]
validating = pml_CSV[-inTrain,]
# number of rows and columns of data in the training set
dim(training)
# number of rows and columns of data in the validating set
dim(validating)
```

```
## [1] 11776   159
## [1] 7846  159
```

Data Exploration and Cleaning

Since we choose a random forest model and we have a data set with too many columns, first we check if we have many problems with columns without data. So, remove columns that have less than 60% of data entered.

```
# Number of cols with less than 60% of data
sum((colSums(!is.na(training[,-ncol(training)])) < 0.6*nrow(training)))
```

[1] 100

```
# apply our definition of remove columns that most doesn't have data, before its apply to the model.
Keep <- c((colSums(!is.na(training[,-ncol(training)])) >= 0.6*nrow(training)))
training   <-  training[,Keep]
validating <- validating[,Keep]
# number of rows and columns of data in the final training set
dim(training)
```

[1] 11776 59

```
# number of rows and columns of data in the final validating set
dim(validating)
```

[1] 7846 59

Modeling

In random forests, there is no need for cross-validation or a separate test set to get an unbiased estimate of the test set error. It is estimated internally, during the execution. So, we proceed with the training the model (Random Forest) with the training data set.

```
model <- randomForest(classe~.,data=training)
print(model)
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = training)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.2%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3347    1    0    0    0 0.0002986858
## B    2 2277    0    0    0 0.0008775779
## C    0    7 2044    3    0 0.0048685492
## D    0    0    5 1924    1 0.0031088083
## E    0    0    0    4 2161 0.0018475751
```

Model Evaluate

And proceed with the verification of variable importance measures as produced by random Forest:

```
importance(model)
```

```
##                      MeanDecreaseGini
## user_name                 107.1975047
## raw_timestamp_part_1      954.3545884
## raw_timestamp_part_2       10.8921699
## cvtd_timestamp           1396.1802957
## new_window                  0.2786032
## num_window                542.1208347
## roll_belt                 526.9822251
## pitch_belt                284.1486551
## yaw_belt                  341.4011600
## total_accel_belt          112.1259410
## gyros_belt_x               38.5731170
## gyros_belt_y               52.4455886
## gyros_belt_z              119.5443278
## accel_belt_x               58.8222328
## accel_belt_y               74.2622586
## accel_belt_z              172.4890263
## magnet_belt_x             106.1842659
## magnet_belt_y             188.1598221
## magnet_belt_z             179.2909337
## roll_arm                  128.0939580
## pitch_arm                  53.3056872
## yaw_arm                    83.0233335
## total_accel_arm            29.3957265
## gyros_arm_x                44.1855371
## gyros_arm_y                45.3979936
## gyros_arm_z                17.4945953
## accel_arm_x               100.0981311
```

```
## accel_arm_y                53.9169447
## accel_arm_z                39.4375766
## magnet_arm_x              102.6056557
## magnet_arm_y               77.4100148
## magnet_arm_z               51.8083633
## roll_dumbbell             200.8527916
## pitch_dumbbell             81.3157851
## yaw_dumbbell              111.2087833
## total_accel_dumbbell      119.0421160
## gyros_dumbbell_x           43.8862885
## gyros_dumbbell_y          106.4504335
## gyros_dumbbell_z           25.7188985
## accel_dumbbell_x          122.4271753
## accel_dumbbell_y          191.3429701
## accel_dumbbell_z          131.7099197
## magnet_dumbbell_x         237.0838212
## magnet_dumbbell_y         318.3240425
## magnet_dumbbell_z         295.0782199
## roll_forearm              234.6494904
## pitch_forearm             302.7048528
## yaw_forearm                55.5019381
## total_accel_forearm        29.6616488
## gyros_forearm_x            25.2420291
## gyros_forearm_y            41.2936019
## gyros_forearm_z            27.5741559
## accel_forearm_x           136.9632645
## accel_forearm_y            43.2038953
## accel_forearm_z            91.7862461
## magnet_forearm_x           72.9442374
## magnet_forearm_y           75.0993571
## magnet_forearm_z           96.5804487
```

Now we evaluate our model results through confusion Matrix.

```
confusionMatrix(predict(model,newdata=validating[,-ncol(validating)]),validating$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2231    1    0    0    0
##          B    1 1517    3    0    0
##          C    0    0 1364    1    0
##          D    0    0    1 1285    1
##          E    0    0    0    0 1441
##
## Overall Statistics
##
##                Accuracy : 0.999
##                  95% CI : (0.998, 0.9996)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9987
##  Mcnemar's Test P-Value : NA
```

```
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   0.9993   0.9971   0.9992   0.9993
## Specificity            0.9998   0.9994   0.9998   0.9997   1.0000
## Pos Pred Value         0.9996   0.9974   0.9993   0.9984   1.0000
## Neg Pred Value         0.9998   0.9998   0.9994   0.9998   0.9998
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1933   0.1738   0.1638   0.1837
## Detection Prevalence   0.2845   0.1939   0.1740   0.1640   0.1837
## Balanced Accuracy      0.9997   0.9994   0.9985   0.9995   0.9997
```

And confirmed the accuracy at validating data set by calculate it with the formula:

```
accuracy <-c(as.numeric(predict(model,newdata=validating[,-ncol(validating)])==validating$classe))
accuracy <-sum(accuracy)*100/nrow(validating)
```

Model Accuracy as tested over Validation set = **99.9%**.

Model Test

Finally, we proceed with predicting the new values in the testing csv provided, first we apply the same data cleaning operations on it and coerce all columns of testing data set for the same class of previous data set.

Getting Testing Dataset

```
pml_CSV  <- read.csv("C://Users//MAHE//Documents//PMLData//pml-testing.csv", header=TRUE, na.strings=c(
pml_CSV <- pml_CSV[,-1] # Remove the first column that represents a ID Row
pml_CSV <- pml_CSV[ , Keep] # Keep the same columns of testing dataset
pml_CSV <- pml_CSV[,-ncol(pml_CSV)] # Remove the problem ID
# Apply the Same Transformations and Coerce Testing Dataset
# Coerce testing dataset to same class and strucuture of training dataset
testing <- rbind(training[100, -59] , pml_CSV)
# Apply the ID Row to row.names and 100 for dummy row from testing dataset
row.names(testing) <- c(100, 1:20)
```

Predicting with testing dataset

```
predictions <- predict(model,newdata=testing[-1,])
print(predictions)
```

```
##  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
```

get the time

```
endTime <- Sys.time()
```

The analysis was completed on Fri Jan 11 10:52:02 PM 2019 in 1 seconds.