Exercise Prediction

Michael Bristow

14 December 2017

Practical Machine Learning Peer Assessment - Exercide Prediciton

Executive Summary

This report predicts the manner in which users of exercise devices perform the 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: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

Data Used

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

Get data files to use for training and testing

```
#check if data dorectory exists
library(caret)
```

```
## Warning: package 'caret' was built under R version 3.4.3
```

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

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

```
## Warning: package 'ggplot2' was built under R version 3.4.3
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.4.3
```

```
## randomForest 4.6-12
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
set.seed(888)


if(!file.exists("./data")){
  dir.create("./data")
}
#download files if required
trainingFile <- "./data/pml-training.csv"
traininjgURL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"

if (!file.exists(trainingFile)) {
  download.file(traininjgURL, destfile=trainingFile)
}

testingFile <- "./data/pml-testing.csv"
testingURL <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

if (!file.exists(testingFile)) {
  download.file(testingURL, destfile=testingFile)
}
```

Data preparation

Load data

```
trainData <- read.csv("./data/pml-training.csv",header=T,sep=",",na.strings=c("NA",""))
testData <- read.csv("./data/pml-testing.csv",header=T,sep=",",na.strings=c("NA",""))
```

Basic data validation

```
#head(trainData)
dim(trainData)
```

```
## [1] 19622   160
```

```
dim(testData)
```

```
## [1]  20 160
```

```
#nrow(trainData)
```

The training data contains 19622 rows and we will be preicting on the variable “classe”

Data cleaning/pre processing

```
#split into training & validating
trainData <- trainData[,-1] # Remove the first column that represents a ID Row
inTrain <- createDataPartition(y=trainData$classe, p=0.7, list=FALSE)
trainingData <- trainData[inTrain,]
validationData <- trainData[-inTrain,]
```

```
sum((colSums(!is.na(trainingData[,-ncol(trainingData)])) < 0.6*nrow(trainingData)))
```

```
## [1] 100
```

```
# Number of cols with less than 50% of data
Keep <- c((colSums(!is.na(trainingData[,-ncol(trainingData)])) >= 0.6*nrow(trainingData)))
trainingData   <-  trainingData[,Keep]
validationData   <-  validationData[,Keep]
```

Data Modelling

We will be using random forest method

```
fmodel <- randomForest(classe~.,data=trainingData)
fmodel
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = trainingData)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.13%
## Confusion matrix:
##      A    B    C    D    E  class.error
## A 3905    1    0    0    0 0.0002560164
## B    3 2654    1    0    0 0.0015048909
## C    0    5 2390    1    0 0.0025041736
## D    0    0    3 2247    2 0.0022202487
## E    0    0    0    2 2523 0.0007920792
```

Data Evaluation

Variable importance of the model and also produce confusion matrix

```
importance(fmodel)
```

```
##                      MeanDecreaseGini
## user_name                   92.4217967
## raw_timestamp_part_1      1158.9787831
## raw_timestamp_part_2        12.6042576
## cvtd_timestamp            1628.5204925
## new_window                   0.2314084
## num_window                 637.6804911
## roll_belt                  614.4863382
## pitch_belt                 334.1875469
## yaw_belt                   394.2076621
## total_accel_belt           137.7138517
## gyros_belt_x                44.0366942
## gyros_belt_y                58.4309504
## gyros_belt_z               144.9611127
## accel_belt_x                71.5212972
```

```
## accel_belt_y               72.9422743
## accel_belt_z              236.0055494
## magnet_belt_x             137.9789315
## magnet_belt_y             230.2055200
## magnet_belt_z             207.9815028
## roll_arm                  141.7158039
## pitch_arm                  63.7352891
## yaw_arm                   103.7952912
## total_accel_arm            31.3866937
## gyros_arm_x                45.3631996
## gyros_arm_y                49.0328785
## gyros_arm_z                20.9243629
## accel_arm_x               102.1304260
## accel_arm_y                57.7878661
## accel_arm_z                47.2882316
## magnet_arm_x              107.5581641
## magnet_arm_y               88.6533114
## magnet_arm_z               70.5477725
## roll_dumbbell             223.0074086
## pitch_dumbbell             96.6764399
## yaw_dumbbell              135.4746023
## total_accel_dumbbell      144.1475018
## gyros_dumbbell_x           46.1287288
## gyros_dumbbell_y          118.6170446
## gyros_dumbbell_z           25.7025037
## accel_dumbbell_x          138.5439142
## accel_dumbbell_y          215.0626869
## accel_dumbbell_z          150.2526924
## magnet_dumbbell_x         266.6161728
## magnet_dumbbell_y         378.8487837
## magnet_dumbbell_z         341.8279422
## roll_forearm              260.8944951
## pitch_forearm             367.2072481
## yaw_forearm                63.0406385
## total_accel_forearm        38.5898731
## gyros_forearm_x            27.6197292
## gyros_forearm_y            46.1342638
## gyros_forearm_z            28.6559541
## accel_forearm_x           153.9017495
## accel_forearm_y            48.9286684
## accel_forearm_z           113.6001827
## magnet_forearm_x           85.4515795
## magnet_forearm_y           83.7286717
## magnet_forearm_z          116.4730732
```

```
confusionMatrix(predict(fmodel,newdata=validationData[,-ncol(validationData)]),validationData$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    1    0    0    0
##          B    0 1138    1    0    0
##          C    0    0 1025    1    0
##          D    0    0    0  963    1
```

```
##          E    0    0    0    0 1081
##
## Overall Statistics
##
##                Accuracy : 0.9993
##                  95% CI : (0.9983, 0.9998)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9991
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9991   0.9990   0.9990   0.9991
## Specificity            0.9998   0.9998   0.9998   0.9998   1.0000
## Pos Pred Value         0.9994   0.9991   0.9990   0.9990   1.0000
## Neg Pred Value         1.0000   0.9998   0.9998   0.9998   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1638   0.1839
## Detection Rate         0.2845   0.1934   0.1742   0.1636   0.1837
## Detection Prevalence   0.2846   0.1935   0.1743   0.1638   0.1837
## Balanced Accuracy      0.9999   0.9995   0.9994   0.9994   0.9995
```

```
aCC <-c(as.numeric(predict(fmodel,newdata=validationData[,-ncol(validationData)])==validationData$classe

aCC <-sum(aCC)*100/nrow(validationData)
```

Model Accuracy as tested over Validation set = 99.9320306%

Testing the model

We mow test the model on the testing data and print out the predictions

Data must be cleaned in the same way as training data o ensure same datat format

```
testData <- testData[,-1]
testData <- testData[,Keep]
testData <- testData[,-ncol(testData)]
testing <- rbind(trainingData[100, -59] , testData)

predictions <- predict(fmodel,newdata=testing)
predictions
```

```
## 143   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17
##   A   B   A   B   A   A   E   D   B   A   A   B   C   B   A   E   E   A
##  18  19  20
##   B   B   B
## Levels: A B C D E
```