Practical Machine Learning

Prediction Assignment Writeup

Thibaut Lefebvre

8 January 2017

Summary

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, we will use data recorded from accelerometers on the belt, forearm, arm, and dumbbell of 6 participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways.

More information is available from the website http://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

The goal of this project is to predict the manner in which the volunteers did the exercise. The outcome is classified as either "A", "B", "C", "D" or "E".

Preparation

```
library(caret)
```

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

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

```
library(randomForest)
```

```
## 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
```

```r
set.seed(42)
```

Data loading

```r
dataset <- read.csv("pml-training.csv", na.strings = c("NA", "#DIV/0!", ""))
dim(dataset)
```

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

Data cleaning

Remove incomplete columns from the dataset (the ones with NA values) along with some irrelevant variables (columns 1 to 7).

```r
NAcount <- sapply(1:dim(dataset)[2], function(x)sum(is.na(dataset[,x])))
NAcols <- which(NAcount > 0)
dataset <- dataset[,-NAcols]
dataset <- dataset[,-c(1:7)]
dataset$classe <- as.factor(dataset$classe)
dim(dataset)
```

```
## [1] 19622    53
```

```r
head(dataset)
```

```
##   roll_belt pitch_belt yaw_belt total_accel_belt gyros_belt_x gyros_belt_y
## 1      1.41       8.07    -94.4                3         0.00         0.00
## 2      1.41       8.07    -94.4                3         0.02         0.00
## 3      1.42       8.07    -94.4                3         0.00         0.00
## 4      1.48       8.05    -94.4                3         0.02         0.00
## 5      1.48       8.07    -94.4                3         0.02         0.02
## 6      1.45       8.06    -94.4                3         0.02         0.00
##   gyros_belt_z accel_belt_x accel_belt_y accel_belt_z magnet_belt_x
## 1        -0.02          -21            4           22            -3
## 2        -0.02          -22            4           22            -7
## 3        -0.02          -20            5           23            -2
## 4        -0.03          -22            3           21            -6
## 5        -0.02          -21            2           24            -6
## 6        -0.02          -21            4           21             0
##   magnet_belt_y magnet_belt_z roll_arm pitch_arm yaw_arm total_accel_arm
## 1           599          -313     -128      22.5    -161              34
## 2           608          -311     -128      22.5    -161              34
## 3           600          -305     -128      22.5    -161              34
## 4           604          -310     -128      22.1    -161              34
## 5           600          -302     -128      22.1    -161              34
## 6           603          -312     -128      22.0    -161              34
##   gyros_arm_x gyros_arm_y gyros_arm_z accel_arm_x accel_arm_y accel_arm_z
## 1        0.00        0.00       -0.02        -288         109        -123
## 2        0.02       -0.02       -0.02        -290         110        -125
```

```
## 3        0.02       -0.02       -0.02        -289         110        -126
## 4        0.02       -0.03        0.02        -289         111        -123
## 5        0.00       -0.03        0.00        -289         111        -123
## 6        0.02       -0.03        0.00        -289         111        -122
##   magnet_arm_x magnet_arm_y magnet_arm_z roll_dumbbell pitch_dumbbell
## 1         -368          337          516      13.05217      -70.49400
## 2         -369          337          513      13.13074      -70.63751
## 3         -368          344          513      12.85075      -70.27812
## 4         -372          344          512      13.43120      -70.39379
## 5         -374          337          506      13.37872      -70.42856
## 6         -369          342          513      13.38246      -70.81759
##   yaw_dumbbell total_accel_dumbbell gyros_dumbbell_x gyros_dumbbell_y
## 1    -84.87394                   37                0            -0.02
## 2    -84.71065                   37                0            -0.02
## 3    -85.14078                   37                0            -0.02
## 4    -84.87363                   37                0            -0.02
## 5    -84.85306                   37                0            -0.02
## 6    -84.46500                   37                0            -0.02
##   gyros_dumbbell_z accel_dumbbell_x accel_dumbbell_y accel_dumbbell_z
## 1             0.00             -234               47             -271
## 2             0.00             -233               47             -269
## 3             0.00             -232               46             -270
## 4            -0.02             -232               48             -269
## 5             0.00             -233               48             -270
## 6             0.00             -234               48             -269
##   magnet_dumbbell_x magnet_dumbbell_y magnet_dumbbell_z roll_forearm
## 1              -559               293               -65         28.4
## 2              -555               296               -64         28.3
## 3              -561               298               -63         28.3
## 4              -552               303               -60         28.1
## 5              -554               292               -68         28.0
## 6              -558               294               -66         27.9
##   pitch_forearm yaw_forearm total_accel_forearm gyros_forearm_x
## 1         -63.9        -153                 36            0.03
## 2         -63.9        -153                 36            0.02
## 3         -63.9        -152                 36            0.03
## 4         -63.9        -152                 36            0.02
## 5         -63.9        -152                 36            0.02
## 6         -63.9        -152                 36            0.02
##   gyros_forearm_y gyros_forearm_z accel_forearm_x accel_forearm_y
## 1            0.00           -0.02             192             203
## 2            0.00           -0.02             192             203
## 3           -0.02            0.00             196             204
## 4           -0.02            0.00             189             206
## 5            0.00           -0.02             189             206
## 6           -0.02           -0.03             193             203
##   accel_forearm_z magnet_forearm_x magnet_forearm_y magnet_forearm_z
## 1            -215              -17              654              476
## 2            -216              -18              661              473
## 3            -213              -18              658              469
## 4            -214              -16              658              469
## 5            -214              -17              655              473
## 6            -215               -9              660              478
##   classe
```

```
## 1      A
## 2      A
## 3      A
## 4      A
## 5      A
## 6      A
```

Training

Create a partition of the dataset into a training dataset and a test dataset.

```
inTrain <- createDataPartition(y = dataset$classe, p = 0.6, list = FALSE)
training <- dataset[inTrain,]
testing <- dataset[-inTrain,]
```

We want to predict the "classe" variable ("A", "B", "C", "D" or "E") by using all the remaining variables in the dataset. We first tried making predictions by using a single decision tree (for its simplicity) but the accuracy was poor. Thus we switched to the model below which is based on random forests (regarded as a natural combination of many decision trees).

```
model <- randomForest(classe ~ ., data = training, method = 'class')
```

Testing

Evaluate the quality of this model on the previously defined test dataset (for cross validation).

```
pred <- predict(model, newdata = testing, type = 'class')
confusionMatrix(pred, testing$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2231    6    0    0    0
##          B    1 1510    9    0    0
##          C    0    2 1357   14    0
##          D    0    0    2 1271    9
##          E    0    0    0    1 1433
##
## Overall Statistics
##
##                Accuracy : 0.9944
##                  95% CI : (0.9925, 0.9959)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9929
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
```

```
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9996   0.9947   0.9920   0.9883   0.9938
## Specificity            0.9989   0.9984   0.9975   0.9983   0.9998
## Pos Pred Value         0.9973   0.9934   0.9883   0.9914   0.9993
## Neg Pred Value         0.9998   0.9987   0.9983   0.9977   0.9986
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2843   0.1925   0.1730   0.1620   0.1826
## Detection Prevalence   0.2851   0.1937   0.1750   0.1634   0.1828
## Balanced Accuracy      0.9992   0.9966   0.9947   0.9933   0.9968
```

The previous model satisfies 99% accuracy on the testing dataset (with a significance level of 5%).

Predictions

Use this model to predict the outcome of 20 entries.

```
testset <- read.csv("pml-testing.csv", na.strings = c("NA", "#DIV/0!", ""))
NAcount <- sapply(1:dim(testset)[2], function(x)sum(is.na(testset[,x])))
NAcols <- which(NAcount > 0)
testset <- testset[,-NAcols]
testset <- testset[,-c(1:7)]
dim(testset)
```

```
## [1] 20 53
```

```
result <- predict(model, newdata = testset, type='class')
result
```

```
##  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
```

Conclusion

The proposed model, based on random forests, seems relevant enough to perform the required task of prediction.