Quantify activity quality from activity monitors

You!

2023-03-06

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 Source

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

The data for this project come from this source: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har. If you use the document you create for this class for any purpose please cite them as they have been very generous in allowing their data to be used for this kind of assignment.

Data Description

The outcome variable is `classe`, a factor variable with 5 levels. For this data set, participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in 5 different fashions:

- exactly according to the specification (Class A)
- throwing the elbows to the front (Class B)
- lifting the dumbbell only halfway (Class C)
- lowering the dumbbell only halfway (Class D)
- throwing the hips to the front (Class E)

Initial configuration

The initial configuration consists of loading some required packages and initializing some variables.

```r
#Data variables
training.file   <- 'pml-training.csv'
test.cases.file <- 'pml-testing.csv'
training.url    <-
        'http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv'
test.cases.url  <-
        'http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv'

#R-Packages
IscaretInstalled <- require("caret")
```

```
## Loading required package: caret

## Warning: package 'caret' was built under R version 4.2.2

## Loading required package: ggplot2

## Loading required package: lattice
```

```r
if(!IscaretInstalled){
    install.packages("caret")
    library("caret")
    }
IsrandomForestInstalled <- require("randomForest")
```

```
## Loading required package: randomForest

## Warning: package 'randomForest' was built under R version 4.2.2

## randomForest 4.7-1.1

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

##
## Attaching package: 'randomForest'

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

```r
if(!IsrandomForestInstalled){
    install.packages("randomForest")
    library("randomForest")
    }
IsRpartInstalled <- require("rpart")
```

```
## Loading required package: rpart
```

```
if(!IsRpartInstalled){
    install.packages("rpart")
    library("rpart")
    }
IsRpartPlotInstalled <- require("rpart.plot")
```

```
## Loading required package: rpart.plot
```

```
## Warning: package 'rpart.plot' was built under R version 4.2.2
```

```
if(!IsRpartPlotInstalled){
    install.packages("rpart.plot")
    library("rpart.plot")
    }
# Set seed for reproducability
set.seed(1000)
```

Data processing

In this section the data is downloaded and processed. Some basic transformations and cleanup will be performed, so that `NA` values are omitted. Irrelevant columns such as `user_name`, `raw_timestamp_part_1`, `raw_timestamp_part_2`, `cvtd_timestamp`, `new_window`, and `num_window` (columns 1 to 7) will be removed in the subset.

The `pml-training.csv` data is used to devise training and testing sets. The `pml-test.csv` data is used to predict and answer the 20 questions based on the trained model.

```
# Download data
if(!file.exists(training.file)){
        download.file(training.url, training.file)
}

if(!file.exists(test.cases.file)){
        download.file(test.cases.url,test.cases.file )
}

# Load data
#load variable training and testing respectively

training <- read.csv(training.file, na.strings = c("NA","#DIV/0!", ""))
testing <- read.csv(test.cases.file, na.strings = c("NA", "#DIV/0!", ""))

# Clean data
cleanColumnIndex <- colSums(is.na(training))/nrow(training) < 0.95
training <- training[,cleanColumnIndex]

#Verify
#verify that NA are removed correctly

colSums(is.na(training))/nrow(training)
```

```
##                    X            user_name raw_timestamp_part_1
##                    0                    0                    0
## raw_timestamp_part_2       cvtd_timestamp           new_window
##                    0                    0                    0
##           num_window            roll_belt           pitch_belt
##                    0                    0                    0
##             yaw_belt     total_accel_belt         gyros_belt_x
##                    0                    0                    0
##         gyros_belt_y         gyros_belt_z         accel_belt_x
##                    0                    0                    0
##         accel_belt_y         accel_belt_z        magnet_belt_x
##                    0                    0                    0
##        magnet_belt_y        magnet_belt_z             roll_arm
##                    0                    0                    0
##            pitch_arm              yaw_arm      total_accel_arm
##                    0                    0                    0
##          gyros_arm_x          gyros_arm_y          gyros_arm_z
##                    0                    0                    0
##          accel_arm_x          accel_arm_y          accel_arm_z
##                    0                    0                    0
##         magnet_arm_x         magnet_arm_y         magnet_arm_z
##                    0                    0                    0
##        roll_dumbbell       pitch_dumbbell         yaw_dumbbell
##                    0                    0                    0
## total_accel_dumbbell     gyros_dumbbell_x     gyros_dumbbell_y
##                    0                    0                    0
##     gyros_dumbbell_z     accel_dumbbell_x     accel_dumbbell_y
##                    0                    0                    0
##     accel_dumbbell_z    magnet_dumbbell_x    magnet_dumbbell_y
##                    0                    0                    0
##    magnet_dumbbell_z         roll_forearm        pitch_forearm
##                    0                    0                    0
##          yaw_forearm  total_accel_forearm      gyros_forearm_x
##                    0                    0                    0
##      gyros_forearm_y      gyros_forearm_z      accel_forearm_x
##                    0                    0                    0
##      accel_forearm_y      accel_forearm_z     magnet_forearm_x
##                    0                    0                    0
##     magnet_forearm_y     magnet_forearm_z               classe
##                    0                    0                    0
```

```
colSums(is.na(training))
```

```
##                    X            user_name raw_timestamp_part_1
##                    0                    0                    0
## raw_timestamp_part_2       cvtd_timestamp           new_window
##                    0                    0                    0
##           num_window            roll_belt           pitch_belt
##                    0                    0                    0
##             yaw_belt     total_accel_belt         gyros_belt_x
##                    0                    0                    0
##         gyros_belt_y         gyros_belt_z         accel_belt_x
##                    0                    0                    0
##         accel_belt_y         accel_belt_z        magnet_belt_x
```

```
##                    0                    0                    0
##        magnet_belt_y        magnet_belt_z             roll_arm
##                    0                    0                    0
##            pitch_arm              yaw_arm      total_accel_arm
##                    0                    0                    0
##          gyros_arm_x          gyros_arm_y          gyros_arm_z
##                    0                    0                    0
##          accel_arm_x          accel_arm_y          accel_arm_z
##                    0                    0                    0
##         magnet_arm_x         magnet_arm_y         magnet_arm_z
##                    0                    0                    0
##        roll_dumbbell       pitch_dumbbell         yaw_dumbbell
##                    0                    0                    0
## total_accel_dumbbell     gyros_dumbbell_x     gyros_dumbbell_y
##                    0                    0                    0
##     gyros_dumbbell_z     accel_dumbbell_x     accel_dumbbell_y
##                    0                    0                    0
##     accel_dumbbell_z    magnet_dumbbell_x    magnet_dumbbell_y
##                    0                    0                    0
##    magnet_dumbbell_z         roll_forearm        pitch_forearm
##                    0                    0                    0
##          yaw_forearm  total_accel_forearm      gyros_forearm_x
##                    0                    0                    0
##      gyros_forearm_y      gyros_forearm_z      accel_forearm_x
##                    0                    0                    0
##      accel_forearm_y      accel_forearm_z     magnet_forearm_x
##                    0                    0                    0
##     magnet_forearm_y     magnet_forearm_z               classe
##                    0                    0                    0
```

```
# Subset data
training <- training[,-c(1:7)]
testing <- testing[,-c(1:7)]
```

Cross-validation

In this section cross-validation will be performed by splitting the training data in training (75%) and testing (25%) data.

```
subSamples <- createDataPartition(y=training$classe, p=0.75)[[1]]
subTraining <- training[subSamples, ]
subTrainingCrossVal <- training[-subSamples, ]
```

Test Data Processing

Now change the test data set into the same

```
allNames <- names(training)
testing <- testing[,allNames[1:52]]
```

Expected out-of-sample error

The expected out-of-sample error in this project will correspond to [1 - (the accuracy in the cross-validation data)]. Accuracy is the proportion of correctly classified observation over the total sample in the subTesting data set. Expected accuracy is the expected accuracy in the out-of-sample data set (i.e. original testing data set). Thus, the expected value of the out-of-sample error will correspond to the expected number of miss classified observations/total observations in the Test data set, which is [1 - (the accuracy found from the cross-validation data set)].

Prediction models

In this section a decision tree and random forest will be applied to the data.

Decision Tree

```
fitDecisionTree <- train(classe ~ ., method = 'rpart', data = subTraining)
```

Predict with decision tree and output the confusion matrix. It seems like the result of the model is not ideal.

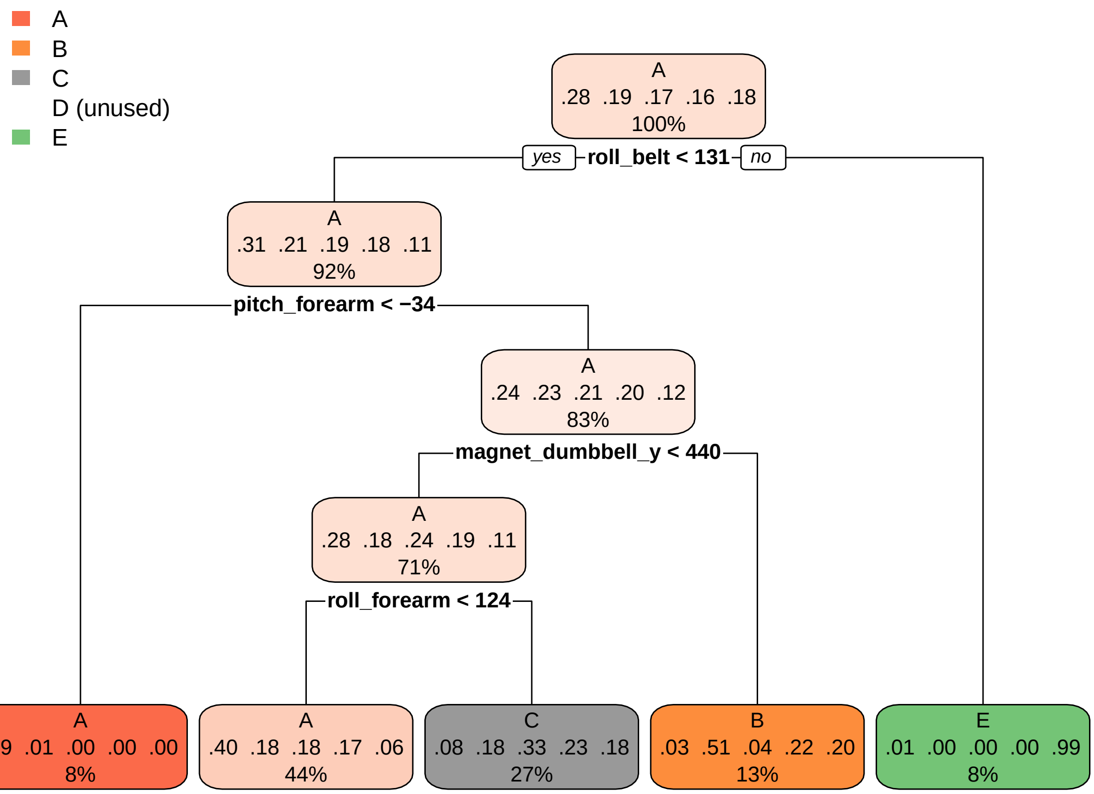
```
predDecisionTree <- predict(fitDecisionTree, subTrainingCrossVal)
confusionMatrix(as.factor(subTrainingCrossVal$classe), predDecisionTree)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1260   24  110    0    1
##          B  393  327  229    0    0
##          C  396   26  433    0    0
##          D  361  149  294    0    0
##          E  118  118  259    0  406
##
## Overall Statistics
##
##                Accuracy : 0.4947
##                  95% CI : (0.4806, 0.5088)
##     No Information Rate : 0.5155
##     P-Value [Acc > NIR] : 0.9983
##
##                   Kappa : 0.34
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.4984  0.50776   0.3268       NA  0.99754
## Specificity            0.9432  0.85399   0.8821   0.8361  0.88993
## Pos Pred Value         0.9032  0.34457   0.5064       NA  0.45061
## Neg Pred Value         0.6386  0.91985   0.7797       NA  0.99975
```

```
## Prevalence             0.5155  0.13132   0.2702   0.0000  0.08299
## Detection Rate         0.2569  0.06668   0.0883   0.0000  0.08279
## Detection Prevalence   0.2845  0.19352   0.1743   0.1639  0.18373
## Balanced Accuracy      0.7208  0.68088   0.6044       NA  0.94373
```

Plot the decision tree

```
rpart.plot(fitDecisionTree$finalModel)
```



Random Forest

```
subTraining$classe <- as.factor(subTraining$classe)
```

Develop training model

```
#fitRandomForest <- train(classe ~ ., method = 'rf', data = subTraining, trControl = fitControl)
fitRandomForest <- randomForest(classe ~ ., data=subTraining, method="class")
predRandomForest <- predict(fitRandomForest, subTrainingCrossVal)
confusionMatrix(predRandomForest, as.factor(subTrainingCrossVal$classe))
```

```
## Confusion Matrix and Statistics
##
```

```
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    6    0    0    0
##          B    0  937    7    0    0
##          C    0    6  848   11    0
##          D    0    0    0  793    6
##          E    0    0    0    0  895
##
## Overall Statistics
##
##                Accuracy : 0.9927
##                  95% CI : (0.9899, 0.9949)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9907
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9874   0.9918   0.9863   0.9933
## Specificity            0.9983   0.9982   0.9958   0.9985   1.0000
## Pos Pred Value         0.9957   0.9926   0.9803   0.9925   1.0000
## Neg Pred Value         1.0000   0.9970   0.9983   0.9973   0.9985
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1911   0.1729   0.1617   0.1825
## Detection Prevalence   0.2857   0.1925   0.1764   0.1629   0.1825
## Balanced Accuracy      0.9991   0.9928   0.9938   0.9924   0.9967
```

Prediction

Now we use it to predict the test set

```
predict(fitRandomForest, testing)
```

```
##  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

As we can see from the result, the random forest algorithem far outperforms the decision tree in terms of accuracy. We are getting 99.25% in sample accuracy, while the decision tree gives us only nearly 50% in sample accuracy