Sport activity quality prediction

Stefano Magarotto

17 agosto 2016

Executive summary

Fitness devices as Jawbone Up, Nike FuelBand and Fitbit allows people to collect a large amount of data about personal activity.

People regularly quantify how much of a particular activity they do, but they rarely quantify how well they do it.

In this project, our goal is to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participants to predict the manner in which they did the exercise. This is the “classe” variable in the training set. We may use any of the other variables to predict with.

We have created a report describing how we built our model, how we used cross validation, what we think the expected out of sample error is, and why we made the choices we did. We will also use our prediction model to predict 20 different test cases.

Data

The training data for this project are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv)

The test data are available here: https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv (https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv)

Preliminary step

In order to obtain reproduceability, we set the seed:

```
set.seed(7000)
```

load the required packages:

```
library(caret)  # to install use: install.packages('caret', dependencies = TRUE)
```

```
## Warning: package 'caret' was built under R version 3.2.5
```

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

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

```
                # so you install also package e1071
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.2.5
```

```
## 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
```

```
library(rpart)
```

```
## Warning: package 'rpart' was built under R version 3.2.5
```

```
library(rpart.plot)
```

```
## Warning: package 'rpart.plot' was built under R version 3.2.5
```

```
library(RColorBrewer)
library(rattle)
```

```
## Warning: package 'rattle' was built under R version 3.2.5
```

```
## Rattle: A free graphical interface for data mining with R.
## Version 4.1.0 Copyright (c) 2006-2015 Togaware Pty Ltd.
## Type 'rattle()' to shake, rattle, and roll your data.
```

get the data:

```
trainUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"
testUrl <- "http://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"
training <- read.csv(url(trainUrl), na.strings=c("NA","#DIV/0!",""))
testing <- read.csv(url(testUrl), na.strings=c("NA","#DIV/0!",""))
```

partition the training set in 2 data sets (60% myTraining, 40% myTest):

```
inTrain <- createDataPartition(y=training$classe, p=0.6, list=FALSE)
myTraining <- training[inTrain, ]; myTest <- training[-inTrain, ]
dim(myTraining); dim(myTest)
```

```
## [1] 11776   160
```

```
## [1] 7846  160
```

clean the data: 1. cleaning NearZeroVariance variables

```
myDataNZV <- nearZeroVar(myTraining, saveMetrics=TRUE)
myNZVvars <- names(myTraining) %in% c("new_window", "kurtosis_roll_belt", "kurtosi
s_picth_belt",
"kurtosis_yaw_belt", "skewness_roll_belt", "skewness_roll_belt.1", "skewness_yaw_b
elt",
"max_yaw_belt", "min_yaw_belt", "amplitude_yaw_belt", "avg_roll_arm", "stddev_roll
_arm",
"var_roll_arm", "avg_pitch_arm", "stddev_pitch_arm", "var_pitch_arm", "avg_yaw_arm
",
"stddev_yaw_arm", "var_yaw_arm", "kurtosis_roll_arm", "kurtosis_picth_arm",
"kurtosis_yaw_arm", "skewness_roll_arm", "skewness_pitch_arm", "skewness_yaw_arm",
"max_roll_arm", "min_roll_arm", "min_pitch_arm", "amplitude_roll_arm", "amplitude_
pitch_arm",
"kurtosis_roll_dumbbell", "kurtosis_picth_dumbbell", "kurtosis_yaw_dumbbell", "ske
wness_roll_dumbbell",
"skewness_pitch_dumbbell", "skewness_yaw_dumbbell", "max_yaw_dumbbell", "min_yaw_d
umbbell",
"amplitude_yaw_dumbbell", "kurtosis_roll_forearm", "kurtosis_picth_forearm", "kurt
osis_yaw_forearm",
"skewness_roll_forearm", "skewness_pitch_forearm", "skewness_yaw_forearm", "max_ro
ll_forearm",
"max_yaw_forearm", "min_roll_forearm", "min_yaw_forearm", "amplitude_roll_forearm"
,
"amplitude_yaw_forearm", "avg_roll_forearm", "stddev_roll_forearm", "var_roll_fore
arm",
"avg_pitch_forearm", "stddev_pitch_forearm", "var_pitch_forearm", "avg_yaw_forearm
",
"stddev_yaw_forearm", "var_yaw_forearm")
myTraining <- myTraining[!myNZVvars]
```

2. remove first column because it can interfer with ML Algorithms

```
myTraining <- myTraining[c(-1)]
dim(myTraining)
```

```
## [1] 11776    99
```

3. remove variables with excessive NAs (> 50%)

```
myTrainingTemp <- myTraining
for(i in 1:length(myTraining)) {
        if( sum( is.na( myTraining[, i] ) ) / nrow(myTraining) >= .5 ) {
        for(j in 1:length(myTrainingTemp)) {
            if( length( grep(names(myTraining[i]), names(myTrainingTemp)[j]) ) ==1
)  {
                myTrainingTemp <- myTrainingTemp[ , -j]
            }
        }
    }
}
dim(myTrainingTemp)
```

```
## [1] 11776    58
```

```
myTraining <- myTrainingTemp
```

4. the same for myTest

```
rem1 <- colnames(myTraining)
rem2 <- colnames(myTraining[, -58])
myTest <- myTest[rem1]
testing <- testing[rem2]
dim(myTest)
```

```
## [1] 7846   58
```

```
dim(testing)
```

```
## [1] 20 57
```

5. coerce the data of the Test data set into the same type

```
for (i in 1:length(testing) ) {
        for(j in 1:length(myTraining)) {
        if( length( grep(names(myTraining[i]), names(testing)[j]) ) ==1)  {
            class(testing[j]) <- class(myTraining[i])
        }
    }
}
```

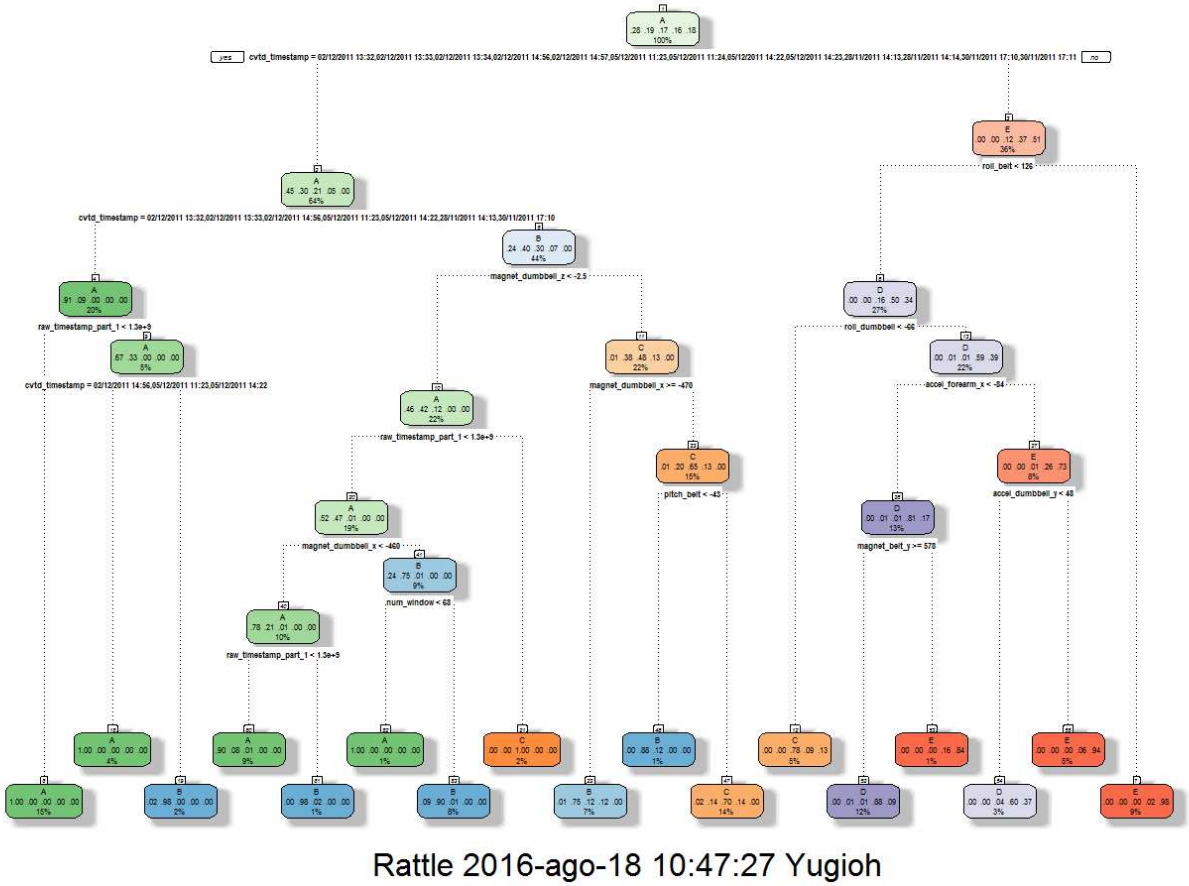
and check it worked

```
testing <- rbind(myTraining[2, -58] , testing)
testing <- testing[-1,]
```

Prediction using Decision tree ML algorithm

Creating and viewing the Decision tree:

```
modFitA1 <- rpart(classe ~ ., data=myTraining, method="class")
fancyRpartPlot(modFitA1)
```



and predicting:

```
predictionsA1 <- predict(modFitA1, myTest, type = "class")
```

test results with confusion matrix:

```
confusionMatrix(predictionsA1, myTest$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2139   61    5    2    0
##          B   67 1254   94   57    0
##          C   26  193 1242  206   61
##          D    0   10   27  967  172
##          E    0    0    0   54 1209
##
## Overall Statistics
##
##                Accuracy : 0.8681
##                  95% CI : (0.8604, 0.8755)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8333
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9583   0.8261   0.9079   0.7519   0.8384
## Specificity            0.9879   0.9655   0.9250   0.9681   0.9916
## Pos Pred Value         0.9692   0.8519   0.7188   0.8223   0.9572
## Neg Pred Value         0.9835   0.9586   0.9794   0.9522   0.9646
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2726   0.1598   0.1583   0.1232   0.1541
## Detection Prevalence   0.2813   0.1876   0.2202   0.1499   0.1610
## Balanced Accuracy      0.9731   0.8958   0.9164   0.8600   0.9150
```

Prediction using Random forests ML algorithm

Creating the Random forest:

```
modFitB1 <- randomForest(classe ~. , data=myTraining)
```

and predicting in-sample error:

```
predictionsB1 <- predict(modFitB1, myTest, type = "class")
```

test results with confusion Matrix:

```
confusionMatrix(predictionsB1, myTest$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 2232    1    0    0    0
##          B    0 1517    1    0    0
##          C    0    0 1364    7    0
##          D    0    0    3 1279    1
##          E    0    0    0    0 1441
##
## Overall Statistics
##
##                Accuracy : 0.9983
##                  95% CI : (0.9972, 0.9991)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9979
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9993   0.9971   0.9946   0.9993
## Specificity            0.9998   0.9998   0.9989   0.9994   1.0000
## Pos Pred Value         0.9996   0.9993   0.9949   0.9969   1.0000
## Neg Pred Value         1.0000   0.9998   0.9994   0.9989   0.9998
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.1933   0.1738   0.1630   0.1837
## Detection Prevalence   0.2846   0.1935   0.1747   0.1635   0.1837
## Balanced Accuracy      0.9999   0.9996   0.9980   0.9970   0.9997
```

As we can see, Random forests gave better results.

Generating the files requested

We use the Random forests that gave better results.

```
predictionsB2 <- predict(modFitB1, testing, type = "class")
```

and generate the files with predictions requested:

```
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)
```