Practical Machine Learning Coursera Project

Massimo Malandra

8 de abril de 2018

### Overview:

This document contains and illustrates the analysis conducted for the final project of the *Practical Machine Learning* course offered by the Johns Hopkins University. The goal of this work is to build a model who is able to predict the manner in which a certain person has done a certain physical exercise, based on a set of variables that register movements, using a series of special devices, as explained in the next paragraph.

### 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. The goal of this project is 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://groupware.les.inf.puc-rio.br/har> (see the section on the Weight Lifting Exercise Dataset). A special thanks goes to the authors of this document who have been very generous in allowing their data to be used for this kind of assignment.

### Data loading and data cleaning:

For this project, two different sets of data have been provided: a training and a test set, that can be downloaded at the respective url indicated above. The first contains 19622 records, while the second only 20; both have 160 variables.

urlTrain <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"  
urlTest <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"  
  
trainingData <- read.csv(url(urlTrain), na.strings = c('','NA'))  
testData <- read.csv(url(urlTest), na.strings = c('','NA'))

head(trainingData)  
str(trainingData) #19622 obs. of 160 variables  
  
head(testData)  
str(testData) #20 obs. of 160 variables

Both the training and test files come with numerous NA values (we have already read the file in such a way that blanks are read as NA too): doing a quick check we see that 100 variables in both files have almost 98% of NAs, hence we will remove them from the analysis. Moreover, having a quick look at the structure of the datasets, it is easy to see how the first 7 variables are not related in any way to body movements (for example: raw\_timestamp\_part\_1, raw\_timestamp\_part\_2, cvtd\_timestamp, etc.). We will remove these variables too, because they are not going to do anything else than adding noise to the model, not bringing any added value to the prediction of the manner in which the movement has been done.

# Checking and removing NA in both train and test  
naCountTrain <- sapply(trainingData, function(x) sum(is.na(x)))  
table(naCountTrain)

## naCountTrain  
## 0 19216   
## 60 100

trainingData <- trainingData[ , naCountTrain == 0]  
  
naCountTest <- sapply(testData, function(x) sum(is.na(x)))  
table(naCountTest)

## naCountTest  
## 0 20   
## 60 100

testData <- testData[ , naCountTest == 0]

str(trainingData)  
# 19622 obs. of 60 variables  
  
str(testData)  
# 20 obs. of 60 variables

# Removing the first 7 variables who don't bring any added value to the purpose of the analysis:  
trainingData <- trainingData[ , 8:60]  
testData <- testData[ , 8:60]

dim(trainingData) # 19622 obs., 53 var.  
dim(testData) # 20 obs., 53 var.  
  
colnames(trainingData) == colnames(testData) # all TRUE except the last one: OK

### Data partition:

Put aside the test set - who will be used only once the final model will be built and selected -, let’s focus now on partitioning the training dataset in 2 different subsets: train (70%) and test (30%) sets again, based on the *classe* variable, who is the target variable of our analysis. The train set consists now of 13737 observations and 53 variables.

# Partitioning the training set:  
library(caret)  
inTrain <- createDataPartition(trainingData$classe, p=0.7,   
 list=FALSE)  
trainSet <- trainingData[inTrain, ]  
testSet <- trainingData[-inTrain, ]

dim(trainSet) # 13737 obs., 53 var.  
dim(testSet) # 5885 obs., 53 var.

### Prediction models:

Considering the type and the nature of the analytical problem we are facing and the corresponding dataset we have at our disposal, we can try to fit two different machine learning algorithms and evalutate both of them, before deciding which one to select and apply to the test set, we have put aside to actually put our model in action.

#### Decision Tree:

First of all, we will fit a *decision tree* model. The seed will be set at 12345 for reproducibility purposes.

# Decision Tree model:  
library(caret)  
library(rpart)  
set.seed(12345)  
  
decTreeModel <- rpart(classe ~ ., data = trainSet, method = "class")  
  
predDecTree <- predict(decTreeModel, newdata = testSet,   
 type = "class")  
confDecTree <- confusionMatrix(predDecTree, testSet$classe)  
confDecTree

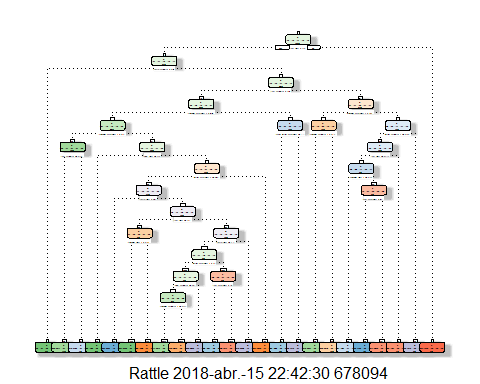
## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 1489 189 21 54 22  
## B 59 680 105 96 125  
## C 45 153 841 124 136  
## D 58 91 59 642 75  
## E 23 26 0 48 724  
##   
## Overall Statistics  
##   
## Accuracy : 0.7436   
## 95% CI : (0.7322, 0.7547)  
## No Information Rate : 0.2845   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.6751   
## Mcnemar's Test P-Value : < 2.2e-16   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 0.8895 0.5970 0.8197 0.6660 0.6691  
## Specificity 0.9321 0.9189 0.9057 0.9425 0.9798  
## Pos Pred Value 0.8389 0.6385 0.6474 0.6941 0.8819  
## Neg Pred Value 0.9550 0.9048 0.9597 0.9351 0.9293  
## Prevalence 0.2845 0.1935 0.1743 0.1638 0.1839  
## Detection Rate 0.2530 0.1155 0.1429 0.1091 0.1230  
## Detection Prevalence 0.3016 0.1810 0.2207 0.1572 0.1395  
## Balanced Accuracy 0.9108 0.7579 0.8627 0.8042 0.8245

The figure below describes the trees with all of its nodes and splits.

library(RColorBrewer)  
library(rattle)

## Rattle: A free graphical interface for data science with R.  
## Versión 5.1.0 Copyright (c) 2006-2017 Togaware Pty Ltd.  
## Escriba 'rattle()' para agitar, sacudir y rotar sus datos.

fancyRpartPlot(decTreeModel)



#### Random Forest:

As second option, we will try to fit a *random forest* model, with the default parameter of 500 as number of trees to grow. In order to avoid that our model would adjust too much to the twirks of our train set - and hence pick the random noise that it carries - we will use a *K-fold cross validation*, with 3 folds. This means that we will use 3 different pairs of train and test, calculating the corresponding errors. The average of these 3 errors will then represent a better estimate of the error we would get in an *out of sample* procedure.

# Random Forest model:  
library(randomForest)  
set.seed(12345)  
  
controlRf = trainControl(method = "cv", number = 3, verboseIter = FALSE)  
randForestModel <- randomForest(classe ~ ., data = trainSet,  
 trControl = controlRf)  
  
predRandForest <- predict(randForestModel, testSet,  
 type = "class")  
confusionMatrix(predRandForest, testSet$classe)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction A B C D E  
## A 1674 7 0 0 0  
## B 0 1129 4 0 0  
## C 0 3 1022 3 0  
## D 0 0 0 961 6  
## E 0 0 0 0 1076  
##   
## Overall Statistics  
##   
## Accuracy : 0.9961   
## 95% CI : (0.9941, 0.9975)  
## No Information Rate : 0.2845   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.9951   
## Mcnemar's Test P-Value : NA   
##   
## Statistics by Class:  
##   
## Class: A Class: B Class: C Class: D Class: E  
## Sensitivity 1.0000 0.9912 0.9961 0.9969 0.9945  
## Specificity 0.9983 0.9992 0.9988 0.9988 1.0000  
## Pos Pred Value 0.9958 0.9965 0.9942 0.9938 1.0000  
## Neg Pred Value 1.0000 0.9979 0.9992 0.9994 0.9988  
## Prevalence 0.2845 0.1935 0.1743 0.1638 0.1839  
## Detection Rate 0.2845 0.1918 0.1737 0.1633 0.1828  
## Detection Prevalence 0.2856 0.1925 0.1747 0.1643 0.1828  
## Balanced Accuracy 0.9992 0.9952 0.9974 0.9978 0.9972

Looking at the accuracy of the two models presented, we can see how the *Random Forest* is the one that allows to reach a higher figure (0.7315 decision tree vs. 0.9962 random forest), hence it is the model that will be selected to be used to predict the type of movement of the 20 observations included in the test data.

### Scoring predictions:

The selected model is now used on the test data, in order to predict the 20 manners in which the movement has been done, indicated with letters from A to E.

# Random Forest prediction:  
randForPred <- predict(randForestModel, testData, type = "class")  
randForPred

## 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