Practical Machine Learning - Course Project - Prediction Assignment

Jérôme Dauge

21 december 2016

Introduction

This report is part of the *Practical Machine Learning* course from Coursera.

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 (see the section on the Weight Lifting Exercise Dataset).

The goal of this project is to predict the manner in which they did the execise.

Datasets used in this project can be found here : training data and test data.

Setup

Two packages are needed in this project :

- caret
- randomForest

The seed is fixed to 42 in order to recreate the results.

```
set.seed(42)

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
```

Loading

The training data and test data are downloaded from the internet to the local workspace. Once they have been downloaded, they are readed and loaded to a data frame.

In the read_csv function, the Not A Number (NaN) representation is specified. They have been defined after a quick look at the data.

```
url_training_file = 'https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv'
url_testing_file = 'https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv'

if(!file.exists("./training.csv")){
  download.file(url=url_training_file, destfile="./training.csv", method="curl")
}
if(!file.exists("./testing.csv")){
  download.file(url=url_testing_file, destfile="./testing.csv", method="curl")
}

nan_representation = c("NA","#DIV/0!","")
training_df = read.csv("./training.csv", na.strings=nan_representation)
testing_df = read.csv("./testing.csv", na.strings=nan_representation)
```

Cleaning

Now that we have load the data, the training and test data should be cleaned.

We will first remove all the columns with nan inside them.

There is a lot of information not necessary in order to build the model.

- For example the user name, the x and all time fields. They are the seven columns to avoid.
- The problem id has also to be avoid in training data frame.

```
non_nullable_columns = names(testing_df[, colSums(is.na(testing_df)) == 0])

selected_features = non_nullable_columns[8:length(non_nullable_columns) - 1]

training_df = training_df[, c(selected_features, 'classe')]
testing_df = testing_df[, c(selected_features, 'problem_id')]
```

Partition

We partionate the data in a training and a testing data set. The data are split with following rule : 75/25 partition.

These partition will allows us to evaluate the sample of error on our predictor.

```
training_partition = createDataPartition(training_df$classe, p=0.75, list=FALSE)
training = training_df[training_partition, ]
testing = training_df[-training_partition, ]
```

Random Forest model

Due to it's accuracy, a random forest model is chosen to estimate the sample. But we will have a focus on the overfitting of the model on the test dataset.

On the testing dataframe, the accuracy should be al least 90% and the error estimate to less than 5 %.

```
decision_tree_model = randomForest(classe~., data=training)
```

Prediction to have Accuracy

Now that the model is built, we would like to know its accuracy and knowing if we need more complex algorithm to match our data.

We will also have a look to the accuracy and see if it match our percentage.

```
prediction = predict(decision_tree_model, newdata=testing)
confusion = confusionMatrix(prediction, testing$classe)
confusion$overall
```

```
##       Accuracy          Kappa  AccuracyLower  AccuracyUpper   AccuracyNull
##      0.9963295      0.9953571      0.9942053      0.9978232      0.2844617
## AccuracyPValue  McnemarPValue
##      0.0000000            NaN
```

The accuracy is about 99.6% which is quite ok. No other algorithm will be explored.

Final result on testing file

A last prediction is made on the testing dataframe provided.

```
prediction_df = predict(decision_tree_model, newdata=testing_df)
prediction_df
```

```
##  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
```