Practical ML - Project Course

Luiz C Pellegrini

March 4, 2016

Coursera Practical Machine Learning Project

March 4th, 2016

Background and Introduction

Using devices such as Jawbone Up, Nike FuelBand, and Fitbit 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 be to use data from accelerometers on the belt, forearm, arm, and dumbell of 6 participant They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. The five ways are 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) and throwing the hips to the front (Class E). Only Class A corresponds to correct performance. The goal of this project is to predict the manner in which they did the exercise, i.e., Class A to E. 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).

The algorithms that will be considered in this project are **classification trees** and **random forests** with the mission to best predict the outcome **classe**. The best performer on the training set will be executed on the test set.

Data Processing

Load the R packages that will be used during this project.

```
    library(caret)
```

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

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

```
    library(rattle)
```

```
## 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.
```

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

```
    library(repmis)
```

Read the training and testing datasets and ignore the null fields

```
    training <- read.csv("pml-training.csv", na.strings = c("NA", ""))
    testing <- read.csv("pml-testing.csv", na.strings = c("NA", ""))
    dim(training)
```

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

```
    dim(testing)
```

```
## [1]  20 160
```

The training dataset has 19622 observations and 160 variables; the testing data set has 20 observations. The goal of this project is to predict the outcome **classe** in the training set using several methods to find out which one is the best.

Data cleaning

This step eliminates NULL values from the training and testing sets

```
    training <- training[, colSums(is.na(training)) == 0]
    testing <- testing[, colSums(is.na(testing)) == 0]
```

This step removes the first seven predictors variables considering that they are no relevant for the outcome **classe**.

```
    trainData <- training[, -c(1:7)]
    testData <- testing[, -c(1:7)]
```

After cleaning phase the datasets contains 53 coluns; training set contains 19633 rowns and testing set contains 20 rows.

Data spliting

Following best practices for predicting values using Machine Learning the training set, trainData, will be splited into 2 new datasets, the first one will be used for training with 70% of the initial file and the validation file, valid, will have the remaining 30% of the initial file. The validation file is also knows as cross validation file according to some other publications.

```
##    set.seed(7826)
    set.seed(2544)
    inTrain <- createDataPartition(trainData$classe, p = 0.7, list = FALSE)
    train <- trainData[inTrain, ]
    valid <- trainData[-inTrain, ]
```

Prediction Algorithms

To solve the problem of predicting the **classe** outcome, **classification trees** and **random forests** will be considered in this project. The method that shows better results on the training set, defined by the level of accuracy from the cofusion matrix, will be applied to the test set.

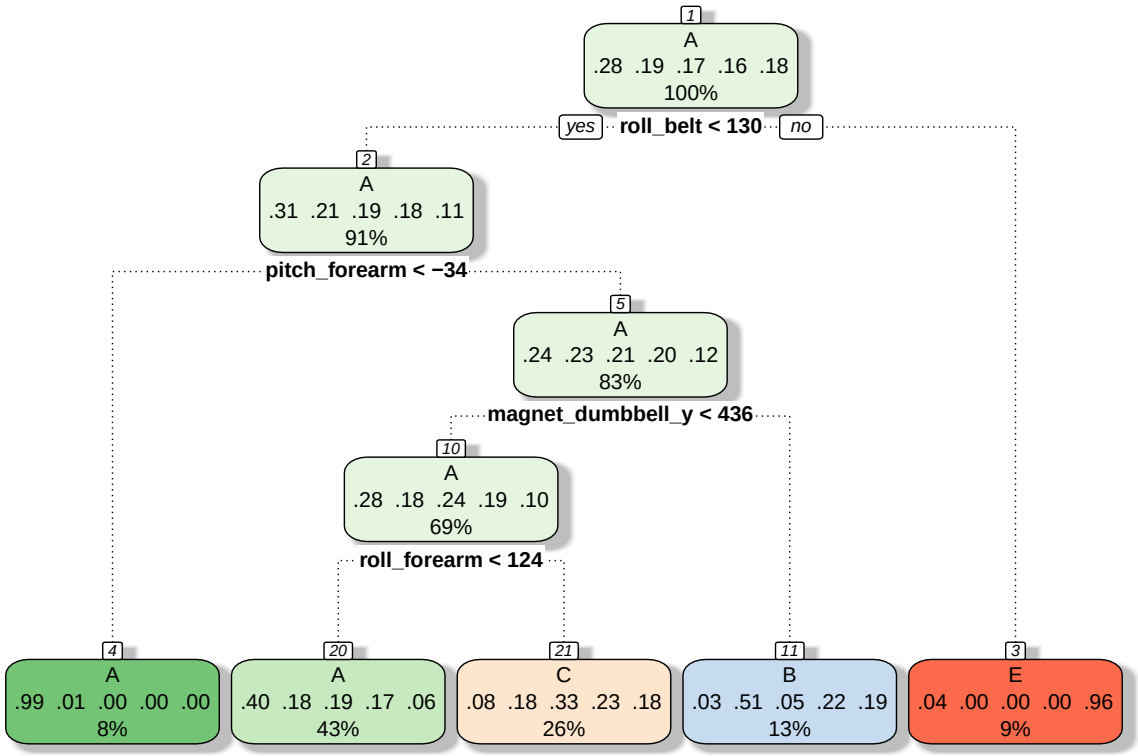
Classification trees

When using k-fold validation with trainControl function the default is set do k=10, but in order to save some computing time k=5 will be considered.

```
    control <- trainControl(method = "cv", number = 5)
    fit_rpart <- train(classe ~ ., data = train, method = "rpart",
                       trControl = control)
    print(fit_rpart, digits = 4)
```

```
## CART
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10989, 10989, 10990, 10989, 10991
## Resampling results across tuning parameters:
##
##   cp       Accuracy  Kappa    Accuracy SD  Kappa SD
##   0.03601  0.4956    0.34122  0.007695     0.01028
##   0.06035  0.4674    0.29498  0.057629     0.09640
##   0.11637  0.3325    0.07361  0.044044     0.06726
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was cp = 0.03601.
```

```
    fancyRpartPlot(fit_rpart$finalModel)
```



Predict outcomes on validation set

```
predict_rpart <- predict(fit_rpart, valid)
```

Prediction result

```
conf_rpart <- confusionMatrix(valid$classe, predict_rpart)
accuracy_rpart <- conf_rpart$overall[1]
```

Accuracy from confusion matrix shows a rate of 0.5 and a out-of-sample rate of 0.5, meaning that classification trees is not a good method to be considered in the **classe** prediction. For this reason we will not run the classifion tree on the test set and we will try the Random Forest.

Random forests

```
fit_rf <- train(classe ~ ., data = train, method = "rf",
                trControl = control)
print(fit_rf, digits = 4)
```

```
## Random Forest
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10989, 10990, 10990, 10990, 10989
## Resampling results across tuning parameters:
##
##   mtry  Accuracy  Kappa   Accuracy SD  Kappa SD
##    2    0.9915    0.9892  0.001139     0.001441
##   27    0.9917    0.9895  0.002729     0.003452
##   52    0.9856    0.9818  0.003726     0.004714
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 27.
```

```
    predict_rf <- predict(fit_rf, valid)
```

Prediction result

```
    conf_rf <- confusionMatrix(valid$classe, predict_rf)
    accuracy_rf <- conf_rf$overall[1]
```

Main conclusions

Random Forest provide a much better result for this training set considering the accuracy rate of 0.991 and the out-of-sample error of 0.009 Random Forest chooses a subset of predictors at each split and decorrelate the trees. This may be the reason of the higher accuracy compared to classification trees, with does not decorrelate predictors. This method takes much longer than the classification trees and for big files this issue can be well addressed before runing the algorithm.

Prediction on Testing Set

After finding a suitable algorithm for the training set the same algorith will be executed with the testing set.

```
    predict(fit_rf, testData)
```

```
##  [1] 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
```