Practical Machine Learning Project

Carlos Castro

10th february 2018

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://groupware.les.inf.puc-rio.br/har (see the section on the Weight Lifting Exercise Dataset).

Data

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://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.

The Goal

The goal of this project is to predict the manner in which people do the exercise using some machine learning algorithms studied. For this, any model will use the classe as variable outcome, and the meaning of 5 leveles that have the variable 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)
- throwing the hips to the front (Class E)

Approach

The steps to develop this project are explained below:

Load the packages and data for the project

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

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

```
## 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.
```

```
## randomForest 4.6-12
```

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

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:rattle':
##
##     importance
```

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

Data analysis and final data (training and testing)

The training set has 19622 observations and 160 variables. Some of the variables have NAs, so the first step was to delete all the columns with missing values. The next step was delete of the initial data the columns 1 to 7, because this variables are inconvenient for our project. This procedure was made to training and testing sets of data, and a validation of columns names beetween both set also was made.

```
# Delete columns with all missing values
training1<-training[,colSums(is.na(training)) == 0]
testing1<-testing[,colSums(is.na(testing)) == 0]
# Delete variables that are irrelevant to our current project:
# user_name, raw_timestamp_part_1, raw_timestamp_part_,2 cvtd_timestamp, new_window, and  num_window
# (columns 1 to 7).
trainingfinal   <-training1[,-c(1:7)]
testingfinal <-testing1[,-c(1:7)]
# Validation between training and testing data
all.equal(colnames(trainingfinal)[1:length(colnames(trainingfinal))-1], colnames(testingfinal)[1:length
```

```
## [1] TRUE
```

```
dim(trainingfinal)
```

```
## [1] 19622    53
```

With this procedure, the final training set have 19622 observations and 53 variables

Data Slicing (Cross validation)

The data obtained in the above step is divided in training set (75% of the data) and testing set(the 25% of remaining data). The machine learning algorimths are trainning with the first data set and tested with the second one. The model that exibith a bigger accuracy is chose in order to preedict de original testing set (with the name testingfinal)

```
# Partition the data so that 75% of the training dataset into training and the remaining 25% to testing
inTrain <- createDataPartition(y=trainingfinal$classe, p=0.75, list=FALSE)
trainingset <- trainingfinal[inTrain,]
testingset <- trainingfinal[-inTrain,]
dim(trainingset)
```

```
## [1] 14718    53
```

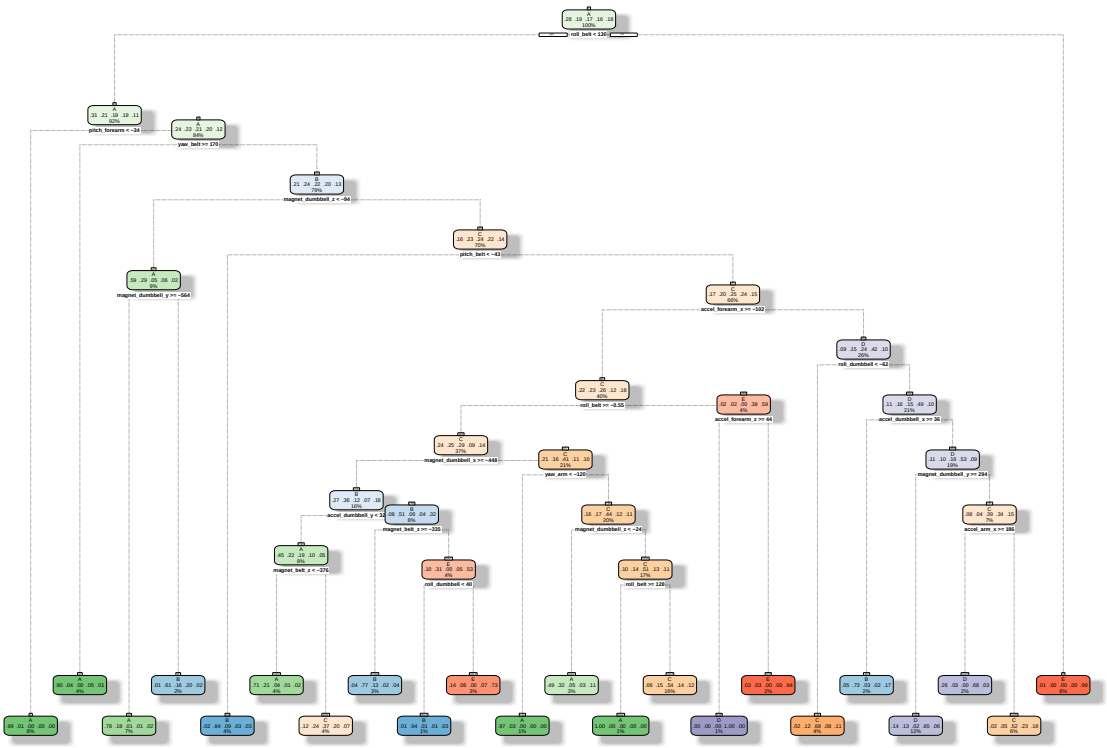
```
dim(testingset)
```

```
## [1] 4904   53
```

ML Model 1: Decision Tree

```
# Prediction with Decision Tree Machine Learning Algorithm
dtmodel<- rpart(classe ~ ., data=trainingset, method="class")
#Plot the Prediction Tree
fancyRpartPlot(dtmodel)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```



Rattle 2018−feb.−11 11:58:40 FACARO

```
# Predictions with Decision Tree
predictionsdt <- predict(dtmodel, testingset, type = "class")
# Evaluate Decision Tree Model Accuracy
confusionMatrix(predictionsdt, testingset$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1180  174   20   30   38
##          B   15  475   61   39   33
##          C   72  213  761  212  187
##          D   97   77   13  506   36
##          E   31   10    0   17  607
##
## Overall Statistics
##
##                Accuracy : 0.7196
##                  95% CI : (0.7068, 0.7322)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6456
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8459  0.50053   0.8901   0.6294   0.6737
## Specificity            0.9253  0.96258   0.8311   0.9456   0.9855
## Pos Pred Value         0.8183  0.76244   0.5266   0.6941   0.9128
## Neg Pred Value         0.9379  0.88928   0.9728   0.9286   0.9306
## Prevalence             0.2845  0.19352   0.1743   0.1639   0.1837
## Detection Rate         0.2406  0.09686   0.1552   0.1032   0.1238
## Detection Prevalence   0.2940  0.12704   0.2947   0.1487   0.1356
## Balanced Accuracy      0.8856  0.73155   0.8606   0.7875   0.8296
```

ML Model 2: Random Forest

```r
# Prediction with Random Forest Machine Learning Algorithm
rfmodel<- randomForest(classe ~ ., data=trainingset)
# Predictions with Random Forest
predictionsrf <- predict(rfmodel, testingset, type = "class")
#Evaluate Random Forest Model Accuracy
confusionMatrix(predictionsrf, testingset$classe)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1395    9    0    0    0
##          B    0  936    1    0    0
##          C    0    4  853    6    1
##          D    0    0    1  798    0
##          E    0    0    0    0  900
##
## Overall Statistics
##
```

```
##                Accuracy : 0.9955
##                  95% CI : (0.9932, 0.9972)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9943
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.9863   0.9977   0.9925   0.9989
## Specificity            0.9974   0.9997   0.9973   0.9998   1.0000
## Pos Pred Value         0.9936   0.9989   0.9873   0.9987   1.0000
## Neg Pred Value         1.0000   0.9967   0.9995   0.9985   0.9998
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2845   0.1909   0.1739   0.1627   0.1835
## Detection Prevalence   0.2863   0.1911   0.1762   0.1629   0.1835
## Balanced Accuracy      0.9987   0.9930   0.9975   0.9961   0.9994
```

Acording with the accuracy, the Random Forest Algorithm shows a 0.9955 accuracy and the Decision Tree Algorithm has an accuracry of 0.7196. So the Random Forest Algorithm is choseen make the predictions of the final testing sample (the accuracy of each model is showed in the confusion Matrix)

Final Predictions and Expected Out-of-Sample Error

The final outcome based on the Random Forest Algorithm, applied to the Final Testing dataset is showed above. For this data, the expected out-of-sample error is estimated at 0.005, or 0.5%, which is obtained from the Confusion Matrix

```
# predict outcome levels on the original Testing data set using Random Forest algorithm
predictionfinal <- predict(rfmodel, testingfinal, type="class")
predictionfinal
```

```
##  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
```

.