Practicial Machine Learning - Course Project

Augustin Jossa

6 novembre 2015

Synopsis

The goal of this project is to predict the manner in which people in the test database did the exercise.

In this report we created a model based on random forest algorythm to evaluate which variables enable us to know how the exercice was done. We then used cross validation to check its accuracy. This model is 98% accurate and its expected sample error is equal to 0.019. We then tested it on the test data base. Here are our results : B A C A A B D B A A B C B A E E A B B B.

The data 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).

The study In this project, we will 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.

The goal of this study is to predict the manner in which the studied people did the exercise. We will thus focus on the "classe" variable.

Copyrights The data for this project comes from http://groupware.les.inf.puc-rio.br/har. We thank them for having been very generous in allowing their data to be used for this study.

Load the data

```
#Set your working directory
setwd("/Users/augustinjossa/Desktop/MOOC/Programming_assignment/Machine/")


#Download the data
if(!file.exists("pml-training")){download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-

if(!file.exists("pml-testing")){download.file("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-


#Read the training data
DT<- read.csv("./pml-training", sep=",", header=TRUE, na.strings = c("NA",""))
DT2<- read.csv("./pml-testing", sep=",", header=TRUE, na.strings = c("NA",""))
```

Clean the data

First, we remove columns with missing values, and those who won't be needed for our analysis.

```
out <- DT[,(colSums(is.na(DT)) == 0)]
out2 <- DT2[, (colSums(is.na(DT2)) == 0)]
outcol <- c("X","user_name","raw_timestamp_part_1","raw_timestamp_part_2","cvtd_timestamp","new_window"

DT_clean<- out[,!(names(out) %in% outcol)]
DT2_clean <- out2[,!(names(out2) %in% outcol)]
dim(DT_clean)
```

```
## [1] 19622    54
```

```
dim(DT2_clean)
```

```
## [1] 20 54
```

We have two new data frames with 19622 observations of 54 variables.

Split the data

We want a 70% observation training dataset to train our model. We will then test it on the last 30%.

```
library(lattice)
library(ggplot2)
library(caret)

set.seed(22519) # For reproducibile purpose
inTrain<- createDataPartition(DT_clean$classe, p=0.70, list=FALSE)
training<- DT_clean[inTrain, ]
testing<- DT_clean[-inTrain, ]
dim(training) ; dim(testing)
```

```
## [1] 13737    54
```

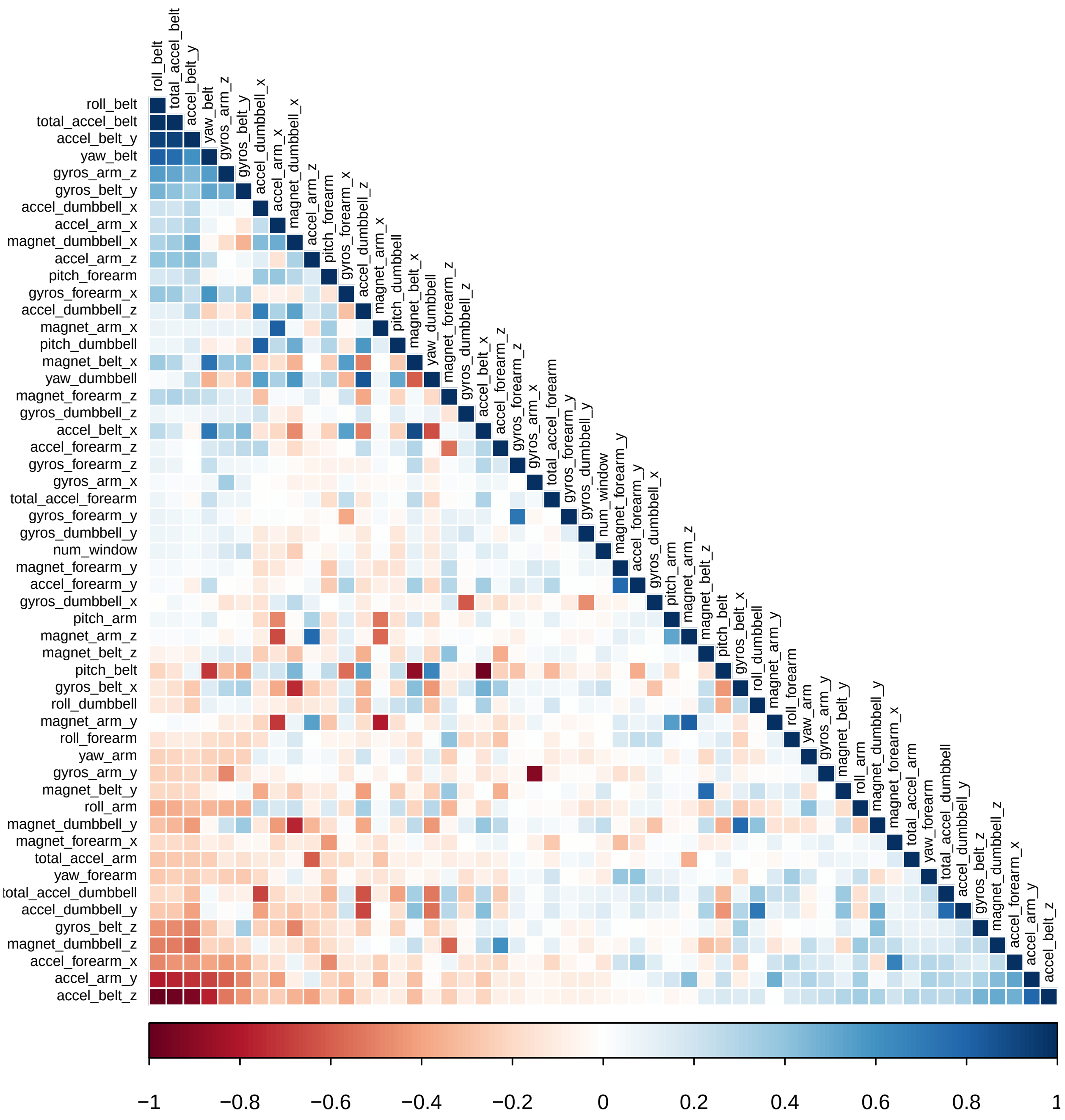
```
## [1] 5885   54
```

We now have a training and a testing database containing 54 variables.Let's see the correlations between theses variables.

Study the correlations between the different variables

```
library(corrplot)
corMatrix<- cor(training[, -54])

corrplot(corMatrix, order = "FPC", method = "color", type = "lower", tl.cex = 0.6, tl.col = rgb(0, 0, 0
```



This graph shows how our columns are correlated. We now use this graph to set a model with uncorrelated variables as our predictors, to increase our accuracy.

Modelisation

We are going to use a random forest algorythm for 4 main reasons :

- It deals automatically with non linearity
- It builds a large number of tree, and thus select more accurate variables
- It eliminates a distinct validation stage, doing it at the same time
- It deals well with outliers variables

Pre-process We pre-process our data using a principal component analysis, leaving out the last column "classe". We then apply the pre-processing to both our training and validation subsets.

```
library(randomForest)
```

```
## randomForest 4.6-12
## Type rfNews() to see new features/changes/bug fixes.
```

```
preProc <- preProcess(training[, -54], method = "pca", thresh = 0.99)
training_preprocess <- predict(preProc, training[, -54])
testing_preprocess <- predict(preProc, testing[, -54])
```

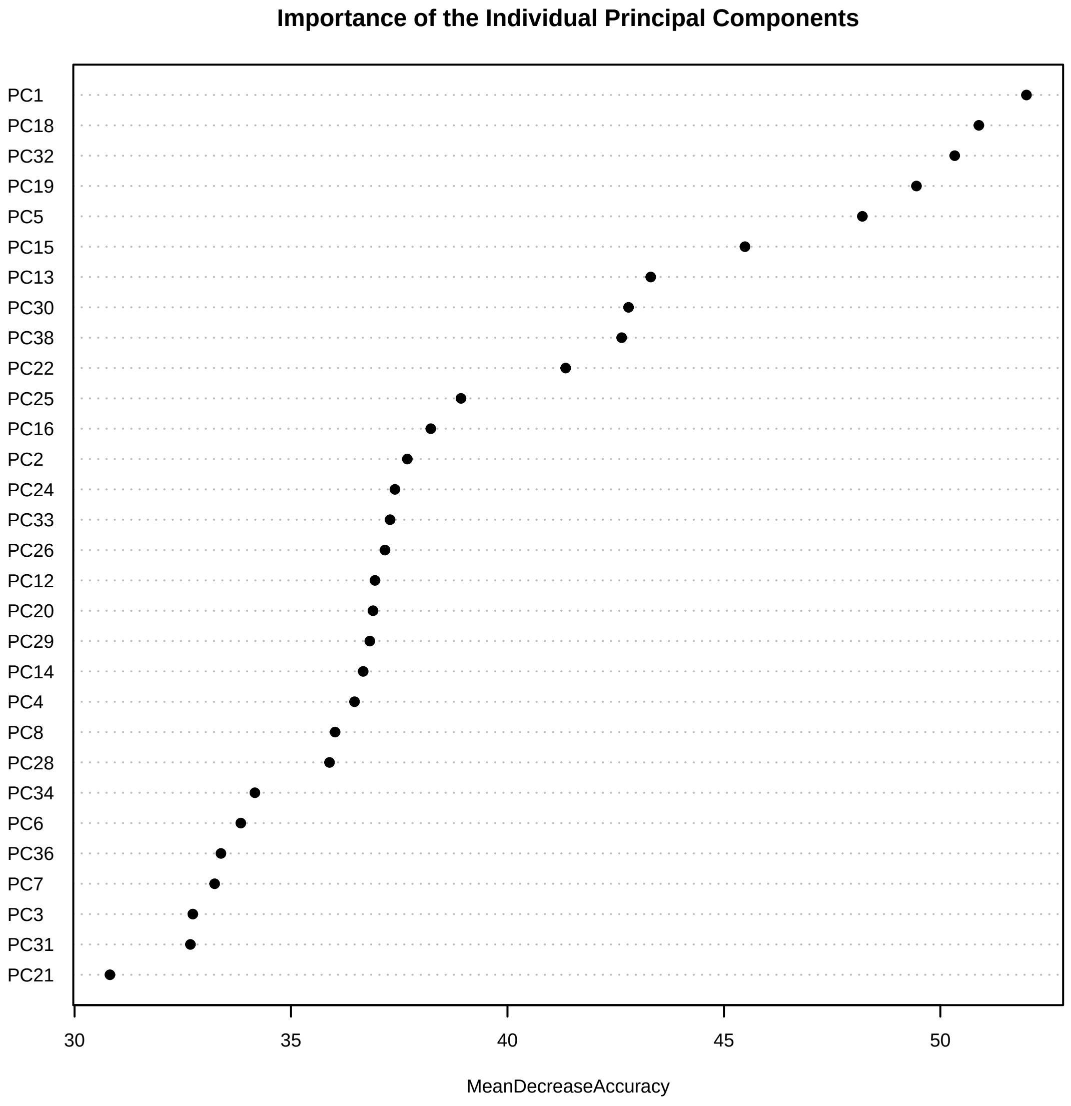
```
library(randomForest)
modFitrf <- train(training$classe ~ ., method = "rf", data = training_preprocess, trControl = trainCont
modFitrf
```

Model

```
## Random Forest
##
## 13737 samples
##    37 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 10989, 10989, 10991, 10990, 10989
## Resampling results across tuning parameters:
##
##   mtry  Accuracy   Kappa      Accuracy SD  Kappa SD
##    2    0.9756861  0.9692249  0.005767474  0.007305015
##   20    0.9708815  0.9631419  0.005305298  0.006721581
##   38    0.9663684  0.9574256  0.006770448  0.008580111
##
## Accuracy was used to select the optimal model using  the largest value.
## The final value used for the model was mtry = 2.
```

Interpretation Let's plot the importance of each individual variable

```
varImpPlot(modFitrf$finalModel, sort = TRUE, type = 1, pch = 19, col = 1, cex = 0.6, main = "Importance
```



This plot shows each of the principal components in order from most important to least important.

Cross Validation Testing and Out-of-Sample Error Estimate

Let's apply our training model on our testing database, to check its accuracy.

```r
predValidRF <- predict(modFitrf, testing_preprocess)
confus <- confusionMatrix(testing$classe, predValidRF)
confus$table
```

Accuracy and Estimated out of sample error

```
##             Reference
## Prediction    A    B    C    D    E
##           A 1670    2    0    1    1
##           B   21 1111    6    0    1
##           C    1   11 1011    3    0
##           D    2    0   48  913    1
##           E    2    2    5    6 1067
```

We can notice that there are very few variables out of this model.

```
accur <- postResample(testing$classe, predValidRF)
modAccuracy <- accur[[1]]
modAccuracy
```

```
## [1] 0.9807986
```

```
out_of_sample_error <- 1 - modAccuracy
out_of_sample_error
```

```
## [1] 0.01920136
```

The estimated accuracy of the model is 98% and the estimated out-of-sample error based on our fitted model applied to the cross validation dataset is 1.9%.

Application of this model on the 20 test cases provided

We have already clean the test data base (DT2_clean). We delete the "problem id" column as it is useless for our analysis.

```
test <- predict(preProc, DT2_clean[, -54])
pred_final <- predict(modFitrf, test)
pred_final
```

```
##  [1] B A C A A B D B A A B C B A E E A B B B
## Levels: A B C D E
```

Here are our results, we will use them for the submission of this course project in the coursera platform.