Machine Learning for Human Activity Recognition

Written by michelg31 - Sunday, December 27, 2015

SUMMARY

Study Background

This document is part of the training Course “Paractical Machine Learning” (from Roger D. Peng, Jeff Leek, and Brian Caffo) that can be found on www.coursera.org.

Assignment

Human Activity Recognition is a field where you try to evaluate activities from series of data captured with wearbale sensors. Goal of the assignmenet is to define a HAR model to quantify how well specific activities are done.

Modelisation results

The modelisation has been built from data collected by Groupware@LES (mailto:Groupware@LES) and bundled under “Weight Lifting Exercises Dataset”. It consists of a data collected on 6 young people asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions (one correct and 4 misdone). Read more information here (http://groupware.les.inf.puc-rio.br/har)

Random Forest modelisation and cross-validation (within training set) approach leads to a very accurate model with a Out-Of-Bag error rate close to 0.67% and an cross-validated accuracy of about 99.44%

DATA EXPLORATION

Data Cleaning

First, we load data and then check consistency around NA’s, empty cells and columns names.Then we clean data from NA’s and empty columns. As a result, only 60 columns out of 160 are kept. We also take out data around participants and time (columns 1 to 7).

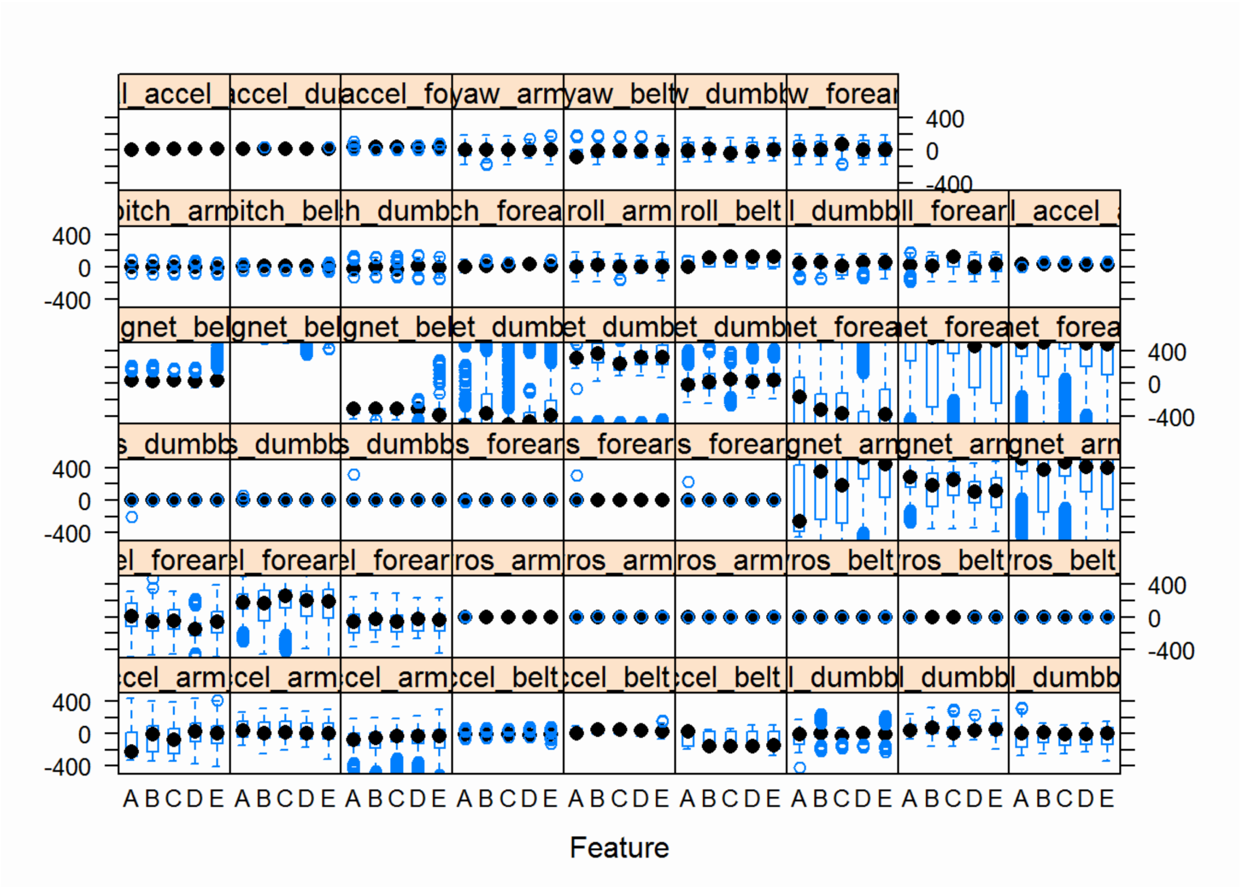
Code to do all these is below :

```r
## checking file and loading data
fileOK <- TRUE
dir <- getwd()
fileNames =c("pml-training.csv", "pml-testing.csv")
for (i in 1:length(fileNames)) if (!any(list.files()==fileNames[i])) fileOK
<- FALSE
if (!fileOK) return(print(paste("Required Files not found in working directo
ry ('",dir,"') ",sep="")))

training <- read.csv(fileNames[1], stringsAsFactors=FALSE)
test <- read.csv(fileNames[2],stringsAsFactors=FALSE)

## checking data
dim(training);summary(training);dim(test);summary(test) ## ckeck data struct
ure
sum(!names(training)==names(test))## check columns names are identicals
dim(training[training$new_window=="no",]);colSums(is.na(training[training$ne
w_window=="yes",]));dim(test[test$new_window=="yes",]) ## NAs analysis

## cleaning data
training <- training[training$new_window=="no",colSums(is.na(training))==0];
test <- test[test$new_window=="no",colSums(is.na(test))==0]  ## eliminate N
A's
training <- training[,colSums(training[training$new_window=="no",]=="")!=dim
(training[training$new_window=="no",])[1],];test <- test[,colSums(test[test
$new_window=="no",]=="")!=dim(test[test$new_window=="no",])[1],] ## eliminat
e unuseful columns
training <- training[,-(1:7)];test <- test[,-(1:7)] ## take out participant
and time independant

## factor classe
library(dplyr)
training <- mutate(training, classe=factor(classe))
```

Plot exploring

Once data has been cleaned, we can take a look on variance between Classe and other variables :

```r
library(caret)
featurePlot(x=training[,1:I(dim(training)[2]-1)], y=training$classe, plot="b
ox", ylim=c(-500,500))
```



```
## more details can been seen with followng command :
## featurePlot(x=training[,grepl("belt",names(training))], y=training$class
e, plot="box")
## featurePlot(x=training[,grepl("_arm",names(training))], y=training$class
e, plot="box")
## featurePlot(x=training[,grepl("dumbbell",names(training))], y=training$cl
asse, plot="box")
## featurePlot(x=training[,grepl("forearm",names(training))], y=training$cla
sse, plot="box")
```

MODELISATION

Since we get enough data, we are going to cross-validate our model, so we need to split training data in a trainingModel dataset to build the model and a trainingValidation dataset to check it's accuracy level on other datasets.

```
set.seed(2567)
train <- createDataPartition(y = training$classe, p = 0.7, list = FALSE)
trainingModel <- training[train, ]
trainingValidation <- training[-train,]
```

From data exploration, modelisation through CART seems more appropriate than linear regression. And, random forest approach seems more appropriate than boosting or bagging. To validate the model, OBB error rate should be lower than 1% and croos-validation accuracy should be above 95%.

Model Buildng

```
library(randomForest)
set.seed(3234)
model <- randomForest(classe ~., data=trainingModel, ntree=100)
model
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = trainingModel, ntree = 100)
##                Type of random forest: classification
##                      Number of trees: 100
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.67%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3826    2    0    1    1 0.001044386
## B   13 2579   11    0    0 0.009220131
## C    0   20 2322    5    0 0.010651896
## D    0    0   23 2179    1 0.010894235
## E    0    1    3    9 2457 0.005263158
```

Error rate is 0.67% which is quite good.

Model cross-validation

```
validationPrediction <- predict(model, trainingValidation)
confusionMatrix(trainingValidation$classe, validationPrediction)
```

```
## Confusion Matrix and Statistics
## 
##           Reference
## Prediction    A    B    C    D    E
##          A 1641    0    0    0    0
##          B    4 1110    1    0    0
##          C    0    9  996    0    0
##          D    0    0   13  929    2
##          E    0    0    0    3 1055
## 
## Overall Statistics
##                                           
##                Accuracy : 0.9944          
##                  95% CI : (0.9922, 0.9962)
##     No Information Rate : 0.2854          
##     P-Value [Acc > NIR] : < 2.2e-16       
##                                           
##                   Kappa : 0.993           
##  Mcnemar's Test P-Value : NA              
## 
## Statistics by Class:
## 
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9976   0.9920   0.9861   0.9968   0.9981
## Specificity            1.0000   0.9989   0.9981   0.9969   0.9994
## Pos Pred Value         1.0000   0.9955   0.9910   0.9841   0.9972
## Neg Pred Value         0.9990   0.9981   0.9971   0.9994   0.9996
## Prevalence             0.2854   0.1942   0.1753   0.1617   0.1834
## Detection Rate         0.2847   0.1926   0.1728   0.1612   0.1831
## Detection Prevalence   0.2847   0.1935   0.1744   0.1638   0.1836
## Balanced Accuracy      0.9988   0.9954   0.9921   0.9968   0.9987
```

Accuracy level has a [99,22% - 99,62%] confidence interval which is also excellent. Model is validated

MODEL PREDICTION

```
testPrediction <- predict(model, test)
testPrediction
```

```
##  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
```