Machine Learning Project - Prediction of barbell lifts

Bruno BERREHUEL*

Abstract
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. 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. This report predicts the manner in which people did the exercises, based on data from accelerometers on the belt, forearm, arm, and dumbell. The detailed results are shown in table **??**, with very good accuracy. In this case, variables with NA aren't necessary to fit a good prediction model. Further fine tuning could be done on the model, but this will not be seen in this report.
A good model should have good accuracy for few computation time. So I recommand using the Bagged Tree model as shown in figure **??**. *It is possible to predict and survey, not only* if *people do exercices, but most of it* how *they do it.*

Keywords
Machine Learning — Predictive Modeling — Monitored fit exercices

***Corresponding author**: METTRE PROFIL LINKED IN

Contents

Introduction

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. 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.

Six candidates were asked to perform barbell lifts correctly and incorrectly in 5 different ways. Results in the datasets are classified in 5 classes, according to how the exercises are done[1] :

- Class A : exactly according to the specification
- Class B : throwing the elbows to the front
- Class C : lifting the dumbbell only halfway
- Class D : lowering the dumbbell only halfway
- Class E : throwing the hips to the front.

1. Dealing with datas

1.1 Exploratory Analysis

There are 160 potentials predictors, for 19622 observations. After a look on the structures of the data presented in appendix, the following columns can be removed because they have no added value for predictive model purpose : V1, user_name, timestamps and windows columns.

There are still 153 potentials predictors, and it seems to have several NAs in the dataset, and they have to be managed.

1.2 Dealing with NA's

There is no data without NA's. So it is necessary to view the distribution of the NA's, and find a solution to deal with them.

min	mean	max	sd
0.9793	0.9811	1.0000	0.0050

Table 1. Summary for mean of NA in predictors with NA

[1] Source : `http://groupware.les.inf.puc-rio.br/har`

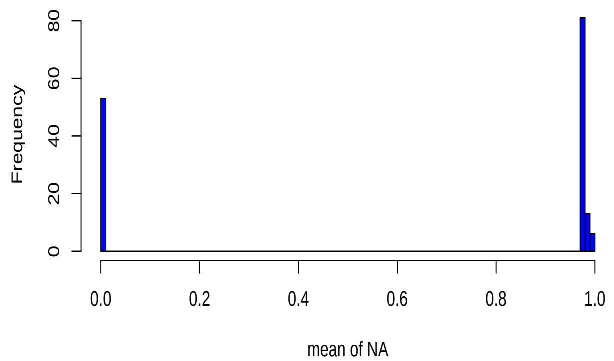

Figure 1. Distribution of NA in predictors

As seen on figure 1 and the table 1, there are only two cases for predictors with NA :

- no NA in the predictor
- a lot of NA in the predictor, more than 97.9%.

Columns can be safely removed in order to reduce noise without available informations

There are now 53 predictors, which will be a little easier to manage with fitting model

2. Fitting a predictive model

The model analysis will be[2] :

1. test variety of models on a small sample of the dataset, with the default tuning. The train will be checked by cross-validation of 10 folds, repeated 3 times, and the distribution of the accuracy, with statistique test, will be reviewed.
2. select the bests 5 of them for accuracy, then train and test them with cross-validation on the complete dataset.
3. choose the best one.

Further analysis could be to tune the selected model in order to increase performance in accuracy and computation time, but it's not the goal of this report.

2.1 Global quick review

The Machine Learning algorithms are choosen to have a good mix of algorithms representations. For this classification case, the 21 models are presented in table 2.

[2]Source : http://machinelearningmastery.com/evaluate-machine-learning-algorithms-with-r/

Bagged Flexible Discriminant Analysis
C5.0
Stacked AutoEncoder Deep Neural Network
Multivariate Adaptive Regression Spline
Stochastic Gradient Boosting
Generalized Partial Least Squares
High Dimensional Discriminant Analysis
Linear Discriminant Analysis
k-Nearest Neighbors
Mixture Discriminant Analysis
Naive Bayes
Neural Network
Parallel Random Forest
Nearest Shrunken Centroids
Penalized Discriminant Analysis
Random Forest
CART
Stabilized Nearest Neighbor Classifier
Support Vector Machines with Linear Kernel
Support Vector Machines with Radial Basis Function Kernel
Bagged CART

Table 2. Models reviewed

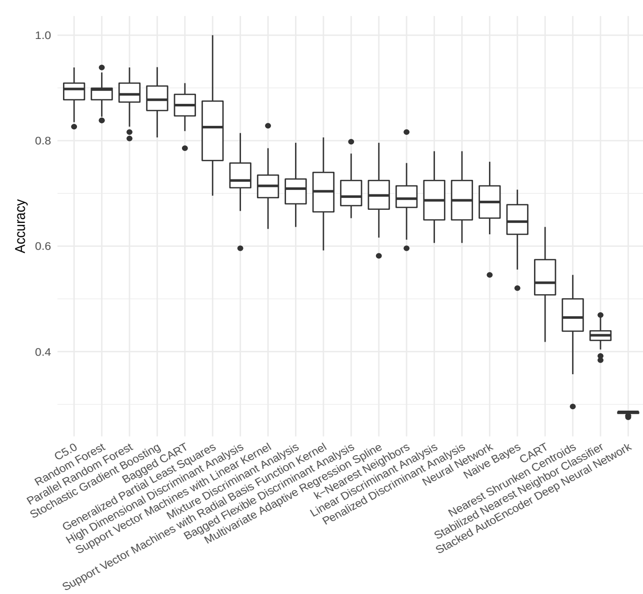

Figure 2. Accuracy of each model on quickview dataset

As seen on figure 2, the five bests models for accuracy are :

C5.0
Random Forest
Parallel Random Forest
Stochastic Gradient Boosting
Bagged CART

The models have pretty good accuracy, from 0.891 to 0.864. The Bagging CART model is especially interesting because of its small computation time, 8.695 seconds, in comparaison of the 81.755 seconds for the C5.0 model. The statistiques tests for the means are significant, as we fail to reject that the true means is not equal to the samples means, so the samples are representatives of the populations. The 95% confidences intervals are presented in table 3.

C5.0	0.8811	0.9018
Random Forest	0.8790	0.8978
Parallel Random Forest	0.8730	0.8977
Stochastic Gradient Boosting	0.8660	0.8891
Bagged CART	0.8535	0.8752

Table 3. Confidences intervals for models accuracy

2.2 Best 5 models study

The 5 bests models will be test on the full dataset, with cross-validation with 10 folders, repeated 3 times.

	Accuracy	Time	method
C5.0	0.997	1454.822	C5.0
Random Forest	0.995	2098.796	rf
Parallel Random Forest	0.995	1042.760	parRF
Bagged CART	0.989	165.225	treebag
Stochastic Gradient Boosting	0.964	656.429	gbm

Table 4. Accuracy and computation time for the 5 bests models

Very very good accuracy, 99%, with small variations, as seen on the figure 3. Small decrease of gbm model, which is now worse than the Bagging CART.

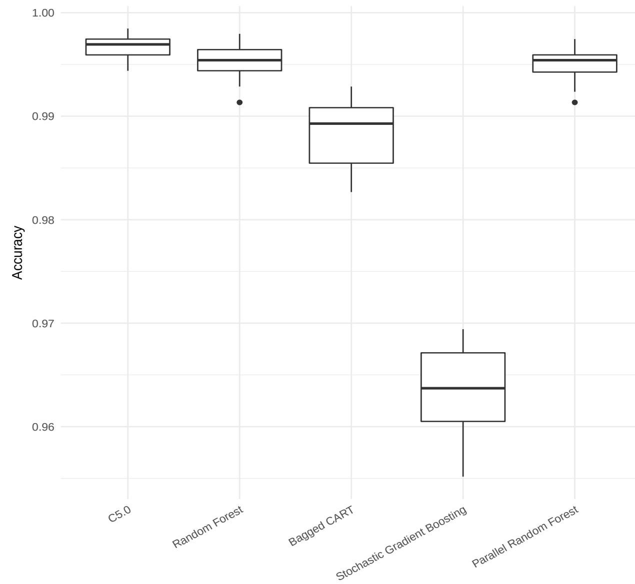

Figure 3. Accuracy of the five best models on total dataset

3. And the winner is...

A good predictive model should be very accurate and very fast.

All the four finalists models are very accurate, so the difference will be in computation time, and the clear winner is here the *Bagged CART Model*, with only 165 seconds of computation, in comparaison of 2099 seconds of the Random Forest, which is the slowest model. The difference appears more clearly on the figure 4.

```
## [1] "Random Forest"
```

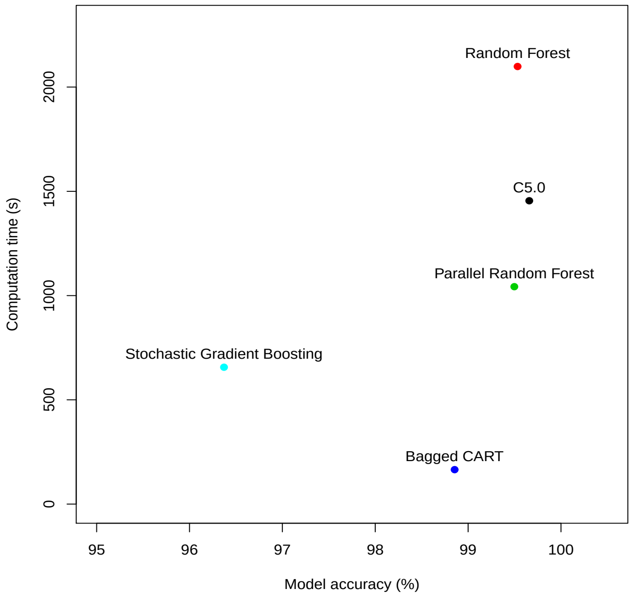

Figure 4. Accuracy vs computation time

Acknowledgments

MERCI POUR SITE RAPPORT ET METHODE : METTRE LES SOURCES EN BAS DE PAGE.

So long and thanks for all the fish [1].

References

[1] A. J. Figueredo and P. S. A. Wolf. Assortative pairing and life history strategy - a cross-cultural study. *Human Nature*, 20:317–330, 2009.

1. Code

A.1 Load Datas

```
library(doMC)
registerDoMC(cores=6) # parallel computation

library(data.table)
library(xtable)

# Downloading files if they don't already exist
if(!file.exists("datas/trainDatas.csv")){
    download.file("https://d396qusza40orc.cloudfront.net/
                predmachlearn/pml-training.csv", "datas/trainDatas.csv")}
if(!file.exists("datas/testDatas.csv")){
    download.file("https://d396qusza40orc.cloudfront.net/
                predmachlearn/pml-testing.csv", "datas/testDatas.csv")}

datas <- fread("datas/trainDatas.csv", na.strings=c("NA","","#DIV/0!"))
quizz <- fread("datas/testDatas.csv", na.strings=c("NA","","#DIV/0!"))
```

A.2 Simplify dataset and deal with NAs

A.2.1 NA distribution

```
# remove unwanted columns
clean1 <- subset(datas, select = c(-1:-7))
# look at NA distribution in predictors
colNA <- apply(clean1, 2, is.na)
meanNA <- apply(colNA, 2, mean)
index <- 1:ncol(colNA)
meanNA <- cbind(meanNA,index)
meanNA <- data.frame(meanNA)

# Plot the distribution
par(mar=c(4,4,1,2))
hist(meanNA$meanNA, main="", breaks=75, col="blue",
     xlab="mean of NA")
```

A.2.2 NA mean

```
# Calculate mean of NA in predictors
naSums <- colSums(is.na(clean1))
dirty <- subset(clean1, select = (naSums!=0))
dirtyColNA <- apply(dirty, 2, is.na)
dirtyMeanNA <- apply(dirtyColNA, 2, mean)
resultsNA <- data.frame(mean=mean(dirtyMeanNA), sd=sd(dirtyMeanNA))
sumNA <- summary(dirtyMeanNA)
mini <- round(min(dirtyMeanNA),4)
maxi <- round(max(dirtyMeanNA),4)
moy <- round(mean(dirtyMeanNA),4)
sdev <- round(sd(dirtyMeanNA),4)
sumNA <- data.frame(min=mini, mean=moy, max=maxi, sd=sdev)
# Remove all predictors with NA from the dataset
# clean is no the reference dataset.
clean <- subset(clean1, select = (naSums==0))

# Print the summary in nice table
print.xtable(xtable(sumNA,
                    caption="Summary for mean of NA in predictors
                    with NA",
                    digits=4, label="namean"), include.rownames=F)
```

A.3 Fitting a model

A.3.1 Model quick review

```
library(caret)
seed <- 27
# create quickview datasets
set.seed(seed)
quickTrain <- createDataPartition(clean$classe, p=0.05)[[1]]
quickview <- clean[quickTrain,]

# functions to automate model test
algo <- function(algo, datas, seed=27, metrics="Accuracy",
                 preprocess=c("center","scale"), controlMeth="repeatedcv",
                 controlNum=10, controlRep=3,...) {
    # function to fit one algorithm  algo on datas, with possibility to choose
    # trainControl, metric, preProc, seed. Datas are centered and scaled by default.
    # add computation time for training at the end of the result.
    require(caret)
    set.seed(seed)
    # set trainControl
    control <- trainControl(method=controlMeth, number=controlNum,
                            repeats=controlRep)
    # set the timer
    ptm <- proc.time()
    fitAlgo <- train(classe~., data=datas, method=algo, metric=metrics,
                     preProc=preprocess, trControl=control,...)
    # record computation time
    timeFit <- proc.time()-ptm
    # store results in a list : x[[y]][[1]] give model,
    # x[[y]][[2]] give computation time
    list(fitAlgo, timeFit[3])
}
```

```
accuAlgo <- function(fitAlgo){
    # research of the best accuracy
    accuracy <- max(fitAlgo[[1]]$results$Accuracy)
    # research of SD for this accuracy
    indAcc <- which.max(fitAlgo[[1]]$results$Accuracy)
    # accuracySD <- fitAlgo[[1]]$results$AccuracySD[indAcc]
    meth <- fitAlgo[[1]]$method
    # store results in data.frame
    res <- data.frame(Accuracy = accuracy, Time=fitAlgo[[2]], method=meth)
    row.names(res) <- fitAlgo[[1]]$modelInfo$label
    res
}

sampleAccu <- function(fitAlgo){
    # store all accuracies computed with the controlTrain in order to boxplot
    # and t.test them
    sampleaccu <- fitAlgo[[1]]$resample$Accuracy
    resu <- data.frame(sampleaccu)
    names(resu) <- fitAlgo[[1]]$modelInfo$label
    resu
}

multiAlgo <- function(multi, datas, seed=27, metrics="Accuracy",
                      preprocess=c("center","scale"),
                      controlMeth="repeatedcv", controlNum=10,
                      controlRep=3){
    # function to test several model and store results in a list
    resultats <- list()
    for (i in multi){
        resultats[match(i,multi)] <- list(algo(datas=datas, algo=i,
                                               seed=seed, metrics=metrics,
                                               preprocess=preprocess,
                                               controlMeth=controlMeth,
                                               controlNum=controlNum,
                                               controlRep=controlRep))
    }
    resultats
}

multiAccu <- function(multi){
    # function to present accuracy, sd and computation time in a ordered dataframe
    # use accuAlgo function to retrieve info for each element of the list
    lapp <- lapply(multi, function(y) accuAlgo(y))
    # loop to store info in table
    leng <- length(multi)
    comput <- NULL
    for (i in 1:leng){
        comput <- rbind(comput,lapp[[i]])
    }
    # order the table on decreasing accuracy
    comput[order(comput$Accuracy, decreasing=T),]
}

multiSampleAccu <- function(multi){
    # function to present the 30 accurracies for each model,
    # in order to plot or t.test them.
    lap <- lapply(multi, function(y) sampleAccu(y))
    len <- length(multi)
    # store in matrix with nrow = nbFold*nbRepeats from trainControl
    compute <- matrix(nrow=nrow(lap[[1]]))
    for (i in 1:len){
        compute <- cbind(compute,lap[[i]])
    }
    compute <- subset(compute, select=c(-1))
    moy <- apply(compute, 2, mean)
    compute <- rbind(compute, moy)
    ordre <- order(compute[31,], decreasing=T)
    compute <- compute[,ordre]
    compute
}
modelAll <- c("bagFDA", "C5.0", "dnn", "earth", "gbm", "gpls", "hdda",
              "lda", "knn", "mda", "nb", "nnet", "parRF", "pam",
              "pda", "rf", "rpart", "snn", "svmLinear", "svmRadial",
              "treebag")

fitAll <- multiAlgo(modelAll, datas=quickview)
orderAllMethod <- as.character(multiAccu(fitAll)$method)

# print the reviewed model in table.
long <- length(fitAll)
model <- NULL
    # faire avec lapply sur list fitAll ?
    model <- sapply(fitAll, function(x) x[[1]]$modelInfo$label)
    model <- data.frame(model)
# for (i in 1:long){
    # # ligne a changer après calcul : fitAll[[i]][[1]]$modelInfo$label
    # model <- rbind(model, fitAll[[i]]$modelInfo$label)
# }
print.xtable(xtable(model, caption="Models reviewed",
                    label="nameTable"), include.rownames=FALSE,
             include.colnames=FALSE, hline.after=c(0,long),
             table.placement="H")

# boxplot all model, oredered by accuracy
library(ggplot2)
plotsAll <- multiSampleAccu(fitAll)
plotsAccu <- multiAccu(fitAll)

library(reshape)
g <- ggplot(data=melt(plotsAll), aes(x=as.factor(variable), y=value))
g <- g + geom_boxplot() + theme_minimal()
g <- g + theme(axis.text.x = element_text(angle=30, hjust=1))
g <- g + ylab("Accuracy") + xlab("")
g

nameFive <- data.frame(row.names(plotsAccu[1:5,]))
print.xtable(xtable(nameFive),include.rownames=F, include.colnames=F,
             hline.after=NULL, table.placement="H")
```

```r
tTest <- apply(plotsAll[,1:5], 2, function(x) t.test(x, mu=mean(x)))
confInt <- sapply(tTest, function(x) x$conf.int)
confInt <- as.data.frame(confInt)
confTable <- matrix(nrow=5, ncol=2)
for (i in 1:5) {
    confTable[i,] <- confInt[,i]
}
row.names(confTable) <- names(plotsAll[,1:5])
print.xtable(xtable(confTable,
                    caption="Confidences intervals for models accuracy",
                    digits=4, label="confInt"), include.colnames=F,
             hline.after=c(0,nrow(confTable)))
```

A.3.2 Best Five models

```r
sampleAccuFive <- multiSampleAccu(fitFive)

g <- ggplot(data=melt(sampleAccuFive), aes(x=as.factor(variable),
                                           y=value))
g <- g + geom_boxplot() + theme_minimal()
g <- g + theme(axis.text.x = element_text(angle=30, hjust=1))
g <- g + ylab("Accuracy") + xlab("")
g

accuTimeFive <- multiAccu(fitFive)
xtable(accuTimeFive, digits=3, label="tableFive",
       caption="Accuracy and computation time for the 5 bests models")
```

A.4 And the Winner is...