REPORT

SUMMARY

Thanks to the dataset provided by HAR (link: http://groupware.les.inf.puc-rio.br/har) I tried to train a model to be able to predict the "classe" feature of a (testing) dataset.

For that purpose, and regarding the very large quantity of data of the training dataset (more than 19,000 rows), I decided to create the following training data repartition : a train set and a test set. My cross-validation method was the "testset validation", were the training data is divided into 2 samples. In this case, each of these sets represent 75% (train set) and 25% (test set) of the training data. I decided to split it in that way due to the large size of the training dataset. Thus, it enables to maximize the accuracy of my future model (large train set) without scarifying a too significant part of the test set.

After cleaning the data from irrelevant features (NAs, empty columns, …), I used the 4 following models, studied in the course:

- Trees model (using the train() function and "rpart" method)

- random Forest model (using the randomForest() function)

- Boosting model (using the train() function and the "gbm" method)

- Bagging model (using the bag() function)

Afterward, I predict the "classe" feature of the test set using each models. Regarding the course, I expected the random Forest model to be the more accurate model/model with the smaller sample error. In order to verify it, I processed the confusion matrix that I divided by the length of the "classe" feature to be able to compute the accuracy (an then the sample error) of each of the 4 models. As I had expected, the random Forest model appeared to be the more accurate one (more than 99.75% accuracy).

Finally, I used the accurate random Forest model I had built to predict the "classe" feature of the testing dataset. It predicts:

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

I commented my R program but you can find further details in the following part of this report. This is the table of content of this second part:

step 0: activation of the required packages
step 1: load the training and the testing datasets
step 2: clear the datasets
step 3: convert factors into integers
step 4: build a train set and a test from the training data
step 5: build some models and use them to predict classes of the test set
step 6: compare the accuracy/error of predictions for each model
step 7: choose the more accurate model and use it to predict the classes of the testing dataset

S0: PACKAGES

The packages I used are: 'caret', 'randomForest' and 'gbm'. I used the `install.packages()` command to be sure that the user is able to activate them.

```
install.packages("gbm")
install.packages("caret")
install.packages("randomForest")

library(gbm)
library(caret)
library(randomForest)
```

S1: LOAD THE DATASETS

Before loading the datasets, I decided to clear workplace.

```
rm(list=ls())
```

The, I loaded the training and testing datasets using their internet link and displayed some of their characteristics (`dim()` and `str()`).

```
# Load the datasets (internet connection required!)
training.data.raw <-
read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
training.csv",header=TRUE,sep=",")
testing.data.raw <-
read.csv("https://d396qusza40orc.cloudfront.net/predmachlearn/pml-
testing.csv",header=TRUE,sep=",")

# Further information on the testing dataset:
dim(testing.data.raw)
str(testing.data.raw)
dim(training.data.raw)
str(training.data.raw)
```

S2: CLEAR THE DATASETS

Thanks to the quick overview of the training and testing datasets, I identified irrelevant columns (time-related columns, empty columns, NAs columns…) that I removed from the raw datasets.

```
# NAs, empty and unsignificant columns cleared
training.data <-
training.data.raw[,c(2,6:11,37:49,60:68,84:86,102,113:124,140,151:160)]
testing.data <-
testing.data.raw[,c(2,6:11,37:49,60:68,84:86,102,113:124,140,151:160)]
```

S3: CONVERT FACTORS INTO INTEGERS

With the aim of being able to train models, I converted factors into integers.

```
classeLevels <- levels(training.data$classe)
training.data <- data.frame(data.matrix(training.data))
training.data$classe <- factor(training.data$classe,labels=classeLevels)
testing.data <- data.frame(data.matrix(testing.data))
```

S4: BUILD A TRAIN AND TEST SETS FROM THE TRAINING DATA

For this work, I set a seed in order to fix the data repartition of this project. Also, as explained above, I build the train set from 75% of the training data and the remaining 25% built the test set.

```
set.seed(2803198)

library(caret)

classeIndex <- which(names(training.data)=="classe")
inTrain <- createDataPartition(y=training.data$classe,p=0.75,list=FALSE)

train.set <- training.data[inTrain,]
test.set <- training.data[-inTrain,]
```

S5: MODELS AND PREDICTIONS

I build 4 models (Trees, random Forest, Boosting, Bagging) I trained on the train set. Then, I used the models to predict, in 4 different ways, the "classe" output of the test set using all the remaining features of the test set.

```
library(caret)

# Trees model
modFit.trees<-train(classe~.,method="rpart",data=train.set)
prediction.trees<-predict(modFit.trees,newdata=test.set)

# Random Forest model
modFit.rf<-randomForest(classe~.,data=train.set,ntree=250)
prediction.rf<-predict(modFit.rf,newdata=test.set)

# Boosting model (/!\ CAN TAKES SEVERAL MINUTES...)
modFit.boost<-train(classe~.,method="gbm",data =
train.set,verbose=FALSE,trControl=trainControl(method="cv",5))
prediction.boost<-predict(modFit.boost,newdata=test.set)

# Bagging model
predictors.bag<-train.set[,-classeIndex]
classe.bag<-train.set$classe
modFit.bag<-bag(predictors.bag,classe.bag,B=10,bagControl=bagControl(fit
= ctreeBag$fit,predict = ctreeBag$pred,aggregate = ctreeBag$aggregate))
prediction.bag<-predict(modFit.bag,newdata = test.set)
```

NOTE: the boosting and bagging models might require some time when run.

S6: ACCURAY/ERROR

I compute the accuracy percentage by using the confusion matrix for each of the 4 models.

```
# Trees model
confusion.matrix.trees<-table(prediction.trees,test.set$classe)
table.trees<-confusion.matrix.trees/length(test.set$classe)
accuracy.trees<-sum(diag(table.trees))

# Random Forest model
confusion.matrix.rf<-table(prediction.rf,test.set$classe)
table.rf<-confusion.matrix.rf/length(test.set$classe)
accuracy.rf<-sum(diag(table.rf))
```

```
# Boosting model
confusion.matrix.boost<-table(prediction.boost,test.set$classe)
table.boost<-confusion.matrix.boost/length(test.set$classe)
accuracy.boost<-sum(diag(table.boost))

# Bagging model
confusion.matrix.bag<-table(prediction.bag,test.set$classe)
table.bag<-confusion.matrix.bag/length(test.set$classe)
accuracy.bag<-sum(diag(table.bag))


# Confusion matrix and tables sum up
# Trees model
confusion.matrix.trees ; table.trees %>% round(digits = 3) ;
c("Accuracy:" , round(accuracy.trees,digits = 5))
# Random Forest model
confusion.matrix.rf ; table.rf %>% round(digits = 3) ; c("Accuracy:" ,
round(accuracy.rf,digits = 5))
# Boosting model
confusion.matrix.boost ; table.boost %>% round(digits = 3) ;
c("Accuracy:" , round(accuracy.boost,digits = 5))
# Bagging model
confusion.matrix.bag ; table.bag %>% round(digits = 3); c("Accuracy:" ,
round(accuracy.bag,digits = 5))
```

With these accuracies, I displayed them in a data frame summing up the accuracy for each on of these 4 models. I also displayed a data frame of the sample error for each model.

```
# Accuracy sum up
accuracy=round(as.numeric(c(accuracy.trees,accuracy.rf,accuracy.boost,acc
uracy.bag)),digits = 4)
sumUp.accuracy=data.frame(accuracy)
rownames(sumUp.accuracy)=c("Trees","random Forest","Boosting","Bagging")
sumUp.accuracy

# Error sum up
sumUp.error=1-sumUp.accuracy
colnames(sumUp.error)=c("error")
sumUp.error
```

S7: PREDICTION

Given the fact the random Forest model shown a higher accuracy (/lower sample error), I chose it to predict the “classe” feature of the testing data.

```
prediction.testing.data<-predict(modFit.rf,newdata=testing.data)
prediction.testing.data
```

The model predicted the following classes for the training dataset:

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