Exercise Prediction

Gonzalo Nelis S.

20/11/2020

Summary

In this report, we build and test a Machine Learning algorithm to predict 'how' certain exercises are performed based on different accelerometer measurements. The data for this assignmet is taken from http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har, specifically on the Weight Lifting Exercise database.

Getting the data

We download the dataset from the given locations:

```r
library(caret)
library(dplyr)
library(data.table)
download.file(url = 'https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv', destfile =
training <- fread('Data/training.csv')

download.file(url = 'https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv', destfile = 'I

testing<-fread('Data/testing.csv')

nfeats<-dim(training)[2]
nobs_training<-dim(training)[1]
nobs_testing<-dim(testing)[1]
```

The training set contains 160 columns and 19622 observations. The testing set is smaller, with only 20 observations. To test the accuracy of our prediction, we will use a validation set extracted from the training set, and we will test the final performance on the testing set.

Cleaning the Data

We perform a data cleaning process to eliminate problematic entries on our training set. A exploratory analysis reveals that there are columns without data, both as zeroes or NAs. Some descriptive variables are also present. We proceed to remove those variables from the data set.

```r
##Delete the first seven columns. They are not related to accelerometer measurements
library(dplyr)
training<-select(training,7:classe)

##Check for variables where there are NA entries
any_na <- function(x) sum(is.na(x))==0
training<- select_if(training,any_na)
```

Now the dataset has 19622 observations and 54 columns.

Model Fitting

We now perform a partition of our dataset to train and validate our model.

```
inTrain<-createDataPartition(y=training$classe, list=FALSE, p = 0.7)

train_set<-training[inTrain,]
validate_set<- training[-inTrain,]
```

Then, we perform the fitting of our model. We will use a Gradient Boosting with Trees and a Random FOrest as they perform generally well for classification problems with a large number of features. To improve performance, we allow for parallel computing following these https://github.com/lgreski/datasciencectacontent/blob/master/markdown/pml-randomForestPerformance.md.

```
library(parallel)
library(doParallel)
cluster <- makeCluster(detectCores() - 1)
registerDoParallel(cluster)
fitControl <- trainControl(method = "cv",
number = 5,
allowParallel = TRUE)

modelgbm<-train(classe~.,data=train_set, method = 'gbm',trControl = fitControl, verbose=FALSE)
modelrf<-train(classe~.,data=train_set, method = 'rf',trControl = fitControl)
```

To assess the quality of our prediction models, we will use the prediction accuracy measure.

```
library(knitr)
cm_rf_train <- confusionMatrix(as.factor(train_set$classe),predict(modelrf,train_set))
cm_gbm_train <- confusionMatrix(as.factor(train_set$classe),predict(modelgbm,train_set))

cm_rf_valid <- confusionMatrix(as.factor(validate_set$classe),predict(modelrf,validate_set))
cm_gbm_valid <- confusionMatrix(as.factor(validate_set$classe),predict(modelgbm,validate_set))

tab<-data.frame(train_set = c(cm_rf_train$overall['Accuracy'],cm_gbm_train$overall['Accuracy']),
                validation_set = c(cm_rf_valid$overall['Accuracy'],cm_gbm_valid$overall['Accuracy']))

rownames(tab)<-c('Random Forest','Gradient Boosting')

kable(tab, caption = 'Accuracy for both models')
```

Table 1: Accuracy for both models

	train_set	validation_set
Random Forest	1.0000000	0.9981308
Gradient Boosting	0.9939579	0.9879354

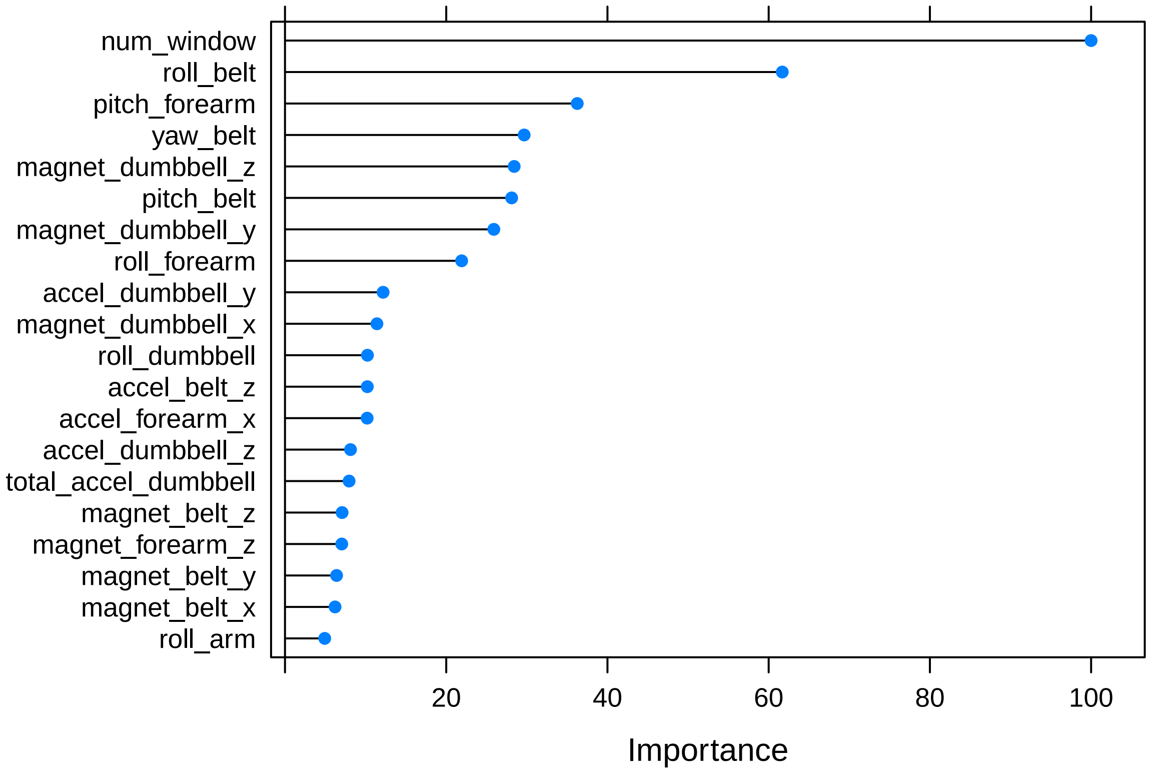
As our results show, gradient boosting is slightly worse than Random Forest, both in the training set and in the validation set. Note the high accuracy measure for the validation set, which reflects a really good prediction quality in our model. We will use the random forest model to predict the testing set labels. Note that the out of sample error (1-accuracy on the validation set) for this model is 0.0018692, which is quite small.

A plot with the 20 most important variables for both models is shown in the following figure:

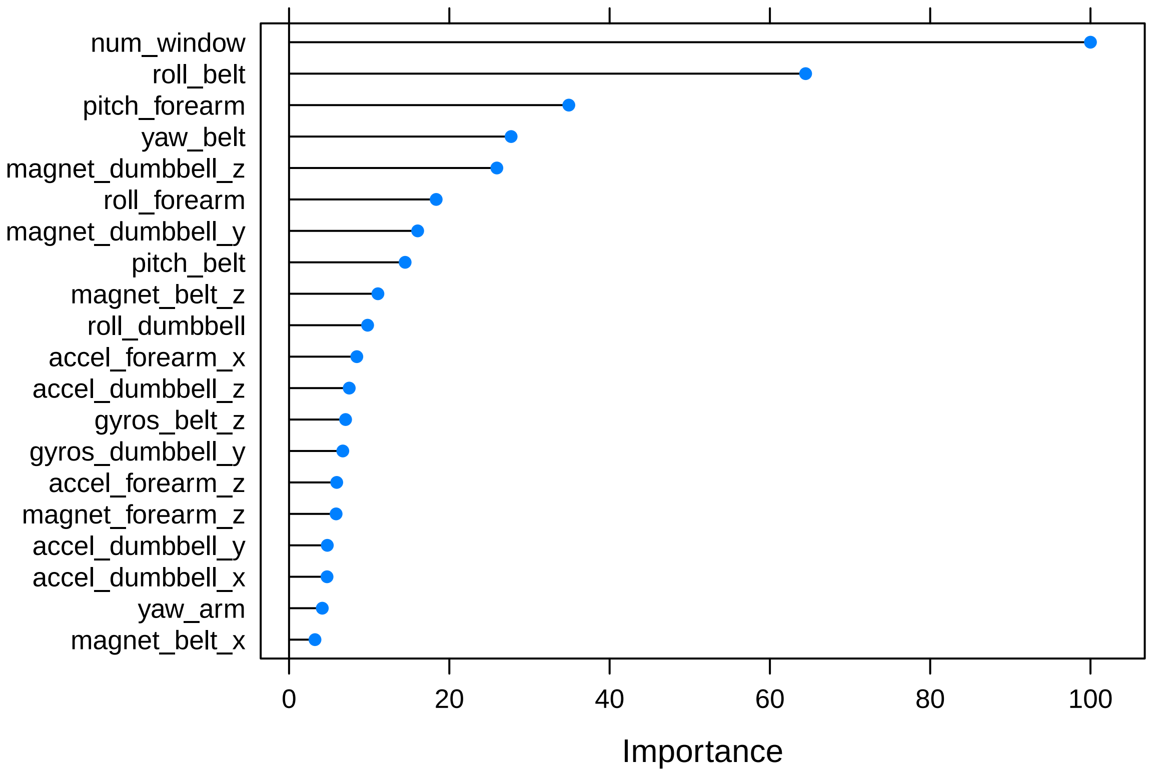
```r
library(gbm)
```

```
## Loaded gbm 2.1.8
```

```r
rfImp <- varImp(modelrf)
gbImp <- varImp(modelgbm)
plot(rfImp, top = 20)
```



```r
plot(gbImp, top = 20)
```



We can see that both models are fairly similar for thew most important features, which explains the similar performance on our accuracy measure.

Prediction on the Testing Set

The final prediction for our random forest is shown next.

```
predict(modelrf,testing)
```

```
##  [1] 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
```