W4 project ML

Gabriel Olivares G

1/5/2021

WEEK 4 MACHINE LEARNING PROJECT ASSIGNMENT

Executive summary

In this document I present the results of the week 4 project assignment. The goal of the project is to estimate a model that can be used to predict the manner in which the participants of an experiment called "Human Activity Recognition" (HAR) have performed the exercise. ("The classe column" in the Data frame).(*)

According to this source the data was collected in the following manner: Six young health participants were asked to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions: Class A: 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; and, Class E: throwing the hips to the front. Class A corresponds to the specified execution of the exercise, while the other 4 classes correspond to common mistakes.

(*) The data for this project come from this source, whose authors kindly made it public –"avaliable) for use"– in the Coursera Machine Learning module: http://web.archive.org/web/20161224072740/http:/groupware.les.inf.puc-rio.br/har

The details of the experiment, the data frame and the metadata of this experiment can be consulted in the following paper: Velloso, E.; Bulling, A.; Gellersen, H.; Ugulino, W.; Fuks, H. Qualitative Activity Recognition of Weight Lifting Exercises. Proceedings of 4th International Conference in Cooperation with SIGCHI (Augmented Human '13) . Stuttgart, Germany: ACM SIGCHI, 2013.

Downloading data

```r
setwd("C:/Users/gog/OneDrive/Documentos/R/COURSERA/Machine learning/w4 project")
library(readr); library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.3     v dplyr   1.0.4
## v tibble  3.0.6     v stringr 1.4.0
## v tidyr   1.1.2     v forcats 0.5.1
## v purrr   0.3.4

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
pml.testing<-as_tibble(
  read.csv("~/R/COURSERA/Machine learning/w4 project/pml-testing.csv"))
pml.training <- as_tibble(
  read.csv("~/R/COURSERA/Machine learning/w4 project/pml-training.csv"))
dim(pml.training);dim(pml.testing)
```

```
## [1] 19622   160
```

```
## [1]  20 160
```

Cleaning data for model estimation: training and test set

```r
sum(colSums(is.na(pml.training))==0)
```

```
## [1] 93
```

```r
sum(colSums(is.na(pml.testing))==0)
```

```
## [1] 60
```

```r
#> Number of complete cases (all variables)
sum(complete.cases(pml.training))
```

```
## [1] 406
```

```r
sum(complete.cases(pml.testing))
```

```
## [1] 0
```

As you may notice, There exists a lot of variables that have missing values. Fortunately this columns seem to be distribution parameters (mean, std, Kurtosis; etc).

Selection of columns to be used

```r
col_names_tr<-names(pml.training)[colSums(is.na(pml.training))>0]
train_clean<-pml.training[,colSums(is.na(pml.training))==0]
col_names_test<-names(pml.testing)[colSums(is.na(pml.testing))>0]
test_clean<-pml.testing[,colSums(is.na(pml.testing))==0]
# As the testing data set has less  variables than the training I selected the
# ones that are common in both data sets.
train_clean<-pml.training[,colSums(is.na(pml.testing))==0]
train_clean$classe<-as.factor(train_clean$classe)
# Clean Data bases
sum(complete.cases(train_clean))
```

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

```r
table(train_clean$user_name,train_clean$classe)
```

```
##
##              A    B    C    D    E
##   adelmo   1165  776  750  515  686
##   carlitos  834  690  493  486  609
##   charles   899  745  539  642  711
##   eurico    865  592  489  582  542
##   jeremy   1177  489  652  522  562
##   pedro     640  505  499  469  497
```

```r
sum(complete.cases(test_clean))
```

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

```r
rm(pml.testing,pml.training)
```

Model estimation

```r
# Model Estimation
library(caret); library(randomForest)
```

```
## Warning: package 'caret' was built under R version 4.0.5

## Loading required package: lattice

##
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':
##
##     lift

## Warning: package 'randomForest' was built under R version 4.0.5

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
## Training and testing data sets for model estimation
set.seed(142857)
inTrain<-createDataPartition(y=train_clean$classe,p=0.7,list=FALSE)
training=train_clean[inTrain,][,8:60]
testing=train_clean[-inTrain,] [8:60]
dim(training); dim(testing)
```

```
## [1] 13737    53
```

```
## [1] 5885   53
```

CART Model (rpart)

```r
ctrl=trainControl(method="cv")
mod_rpart = train(classe~.,training, method="rpart")
print(mod_rpart)
```

```
## CART
##
## 13737 samples
##    52 predictor
##     5 classes: 'A', 'B', 'C', 'D', 'E'
##
## No pre-processing
## Resampling: Bootstrapped (25 reps)
## Summary of sample sizes: 13737, 13737, 13737, 13737, 13737, 13737, ...
## Resampling results across tuning parameters:
##
##   cp          Accuracy   Kappa
##   0.03570339  0.5083287  0.35637157
##   0.06120096  0.4257342  0.22163752
##   0.11463737  0.3323585  0.06939484
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was cp = 0.03570339.
```

```r
# Predict outcomes using the CART model
pred_rpart <- predict(mod_rpart, testing)
# Show prediction result
(confM_rpart <- confusionMatrix(testing$classe, pred_rpart))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1513   28  130    0    3
##          B  474  378  287    0    0
##          C  489   35  502    0    0
##          D  422  179  363    0    0
```

```
##          E  176  150  263    0  493
##
## Overall Statistics
##
##                Accuracy : 0.4904
##                  95% CI : (0.4775, 0.5033)
##     No Information Rate : 0.5223
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.3337
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.4922  0.49091   0.3249       NA  0.99395
## Specificity            0.9427  0.85122   0.8793   0.8362  0.89070
## Pos Pred Value         0.9038  0.33187   0.4893       NA  0.45564
## Neg Pred Value         0.6293  0.91740   0.7853       NA  0.99938
## Prevalence             0.5223  0.13084   0.2625   0.0000  0.08428
## Detection Rate         0.2571  0.06423   0.0853   0.0000  0.08377
## Detection Prevalence   0.2845  0.19354   0.1743   0.1638  0.18386
## Balanced Accuracy      0.7175  0.67107   0.6021       NA  0.94233
```

Random Forest Model

```
mod_rf = randomForest(classe~.,training)
print(mod_rf)
```

```
##
## Call:
##  randomForest(formula = classe ~ ., data = training)
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 7
##
##         OOB estimate of  error rate: 0.52%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 3902    2    2    0    0 0.001024066
## B   13 2643    2    0    0 0.005643341
## C    0   11 2381    4    0 0.006260434
## D    0    0   28 2221    3 0.013765542
## E    0    0    2    5 2518 0.002772277
```

```
# Predict outcomes using the testing set with the Random Forest Model
pred_rf <- predict(mod_rf, testing)
# Prediction result with the Random Forest Model
(confM_rf <- confusionMatrix(testing$classe, pred_rf))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1674    0    0    0    0
##          B    3 1133    3    0    0
##          C    0    5 1019    2    0
##          D    0    0   10  953    1
##          E    0    0    0    0 1082
##
## Overall Statistics
##
##                Accuracy : 0.9959
##                  95% CI : (0.9939, 0.9974)
##     No Information Rate : 0.285
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9948
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9982   0.9956   0.9874   0.9979   0.9991
## Specificity            1.0000   0.9987   0.9986   0.9978   1.0000
## Pos Pred Value         1.0000   0.9947   0.9932   0.9886   1.0000
## Neg Pred Value         0.9993   0.9989   0.9973   0.9996   0.9998
## Prevalence             0.2850   0.1934   0.1754   0.1623   0.1840
## Detection Rate         0.2845   0.1925   0.1732   0.1619   0.1839
## Detection Prevalence   0.2845   0.1935   0.1743   0.1638   0.1839
## Balanced Accuracy      0.9991   0.9972   0.9930   0.9978   0.9995
```

Prediction

According to the accuracy results the "best" model among the two that were estimated is the Random Forest. The accuracy is 99.5% vs. 51% in the CART model The final prediction on the test data frame (validation set?) is the following:

```
predict(mod_rf,test_clean)
```

```
##  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
```