HAR Coursera John´s Hopkins Pratical ML

Samuel Bozzi Baco

0. PACKAGE LOAD AND MARKDOWN CONFIGURATION

```
library(tidyverse)
library(caret)
library(future)
library(doParallel)
library(heatmaply)
library(factoextra)
library(FactoMineR)
library(nnet)
library(future)
library(doParallel)
library(e1071)
```

1. DATA DOWNLOAD

The data will be downloaded using the link from Coursera page.

```
fileLinkTraining <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-training.csv"

fileLinkTest <- "https://d396qusza40orc.cloudfront.net/predmachlearn/pml-testing.csv"

harTraining <- read.csv(fileLinkTraining)

harTest <- read.csv(fileLinkTest)
```

2. DATA WRANGLING

First columns from both dataset (user name, time stamps) will be remove since the model shall be independent from the person who uses it and it is not a time series. Data on test set has several empty columns and "pure NA´s" columns. This ones will be removed from both training set. Columns with no variation at test set will also be removed.

```
not_all_na <- function(x) any(!is.na(x)) # function to determine if the column has all values like NA´s

harTstClean <-
        harTest %>%
        select(-c("X",
                  "user_name",
                  "raw_timestamp_part_1",
```

```
                  "raw_timestamp_part_2",
                  "cvtd_timestamp",
                  "problem_id",
                  "num_window")) %>%
        select_if(not_all_na) %>%
        select_if(~n_distinct(.) > 1)

harTrnClean <- harTraining[ ,c(names(harTstClean), "classe")]
```

The training dataset will be split to allow model test and validation at a proportion to 60/20/20 %.

```
inVal = createDataPartition(harTrnClean$classe, p = 0.2, list = F)

val <- harTrnClean[inVal, ]

model <- harTrnClean[-inVal, ]

inTrain <- createDataPartition(model$classe, p = (0.6/0.8), list = F)

train <- model[inTrain, ]

test <- model[-inTrain, ]
```

3. EDA

3.1 Null model performance

Pre-model tasks are related evaluate the Null Model predictions. This will be accomplished considering the most frequent class in all "predictions", generating a lower limit for any model that will be created, as suggest by Zumel and Mount (2014).

```
nullPred <- test %>% select("classe")

nullPred$pred.class <- names(sort(table(nullPred$classe), decreasing = TRUE)[1])

print(confusionMatrix(as.factor(nullPred$pred.class), reference = as.factor(nullPred$classe)))
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
##          A 1116  759  684  643  721
##          B    0    0    0    0    0
##          C    0    0    0    0    0
##          D    0    0    0    0    0
##          E    0    0    0    0    0
##
## Overall Statistics
##
##                Accuracy : 0.2845
##                  95% CI : (0.2704, 0.2989)
```

```
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : 0.506
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            1.0000   0.0000   0.0000   0.0000   0.0000
## Specificity            0.0000   1.0000   1.0000   1.0000   1.0000
## Pos Pred Value         0.2845      NaN      NaN      NaN      NaN
## Neg Pred Value            NaN   0.8065   0.8256   0.8361   0.8162
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2845   0.0000   0.0000   0.0000   0.0000
## Detection Prevalence   1.0000   0.0000   0.0000   0.0000   0.0000
## Balanced Accuracy      0.5000   0.5000   0.5000   0.5000   0.5000
```

3.2 Covariates correlation

The multicolinearity (covariates correlation) will be investigated, since it can be harmful for some kind of models, like logistic regression. To investigate, a the heatmaply_cor (from package heatmaply) will be user so related covariates will also be grouped together using a hierarchical cluster technique.

```
corrMat <- harTrnClean %>% select(-classe) %>% mutate_if(is.integer, as.numeric) %>% cor()

heatmaply(corrMat, symm = TRUE, cexRow = .0001, cexCol = .0001, branches_lwd = .1)
```

```
## PhantomJS not found. You can install it with webshot::install_phantomjs(). If it is installed, pleas
```

For the plot, it is possible to see that very few covariates presents correlation.

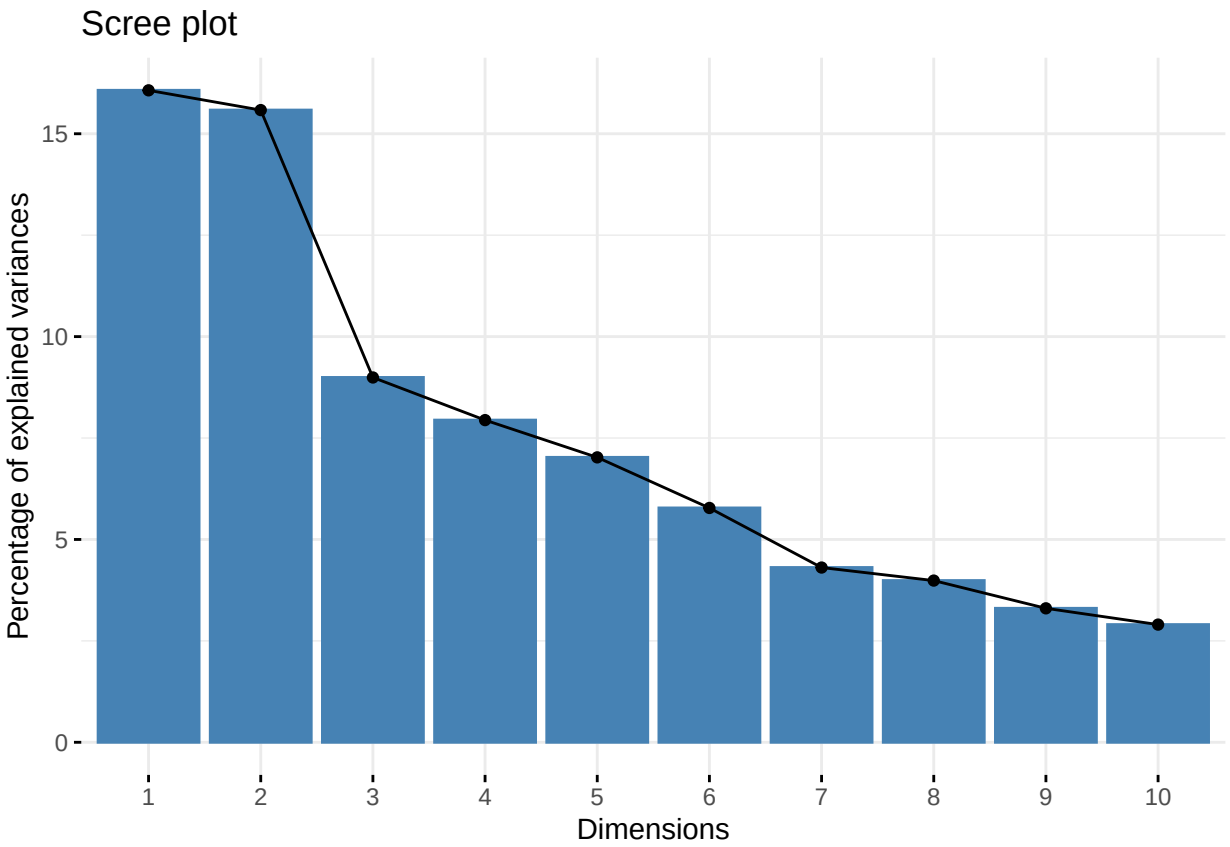
```
pcaCov <- harTrnClean %>% select(-classe) %>% PCA(scale.unit = TRUE, graph = FALSE)

get_eigenvalue(pcaCov)
```

```
##         eigenvalue variance.percent cumulative.variance.percent
## Dim.1  8.356480510      16.07015483                    16.07015
## Dim.2  8.103311777      15.58329188                    31.65345
## Dim.3  4.676019495       8.99234518                    40.64579
## Dim.4  4.129637592       7.94161075                    48.58740
## Dim.5  3.651958340       7.02299681                    55.61040
## Dim.6  3.003559604       5.77607616                    61.38648
## Dim.7  2.239960734       4.30761680                    65.69409
## Dim.8  2.072819572       3.98619149                    69.68028
## Dim.9  1.717230735       3.30236680                    72.98265
## Dim.10 1.508821495       2.90157980                    75.88423
## Dim.11 1.385497930       2.66441910                    78.54865
## Dim.12 1.129241536       2.17161834                    80.72027
## Dim.13 0.986674562       1.89745108                    82.61772
```

```
## Dim.14 0.890702735       1.71288987                 84.33061
## Dim.15 0.836058641       1.60780508                 85.93841
## Dim.16 0.789251336       1.51779103                 87.45620
## Dim.17 0.677935082       1.30372131                 88.75993
## Dim.18 0.609720195       1.17253884                 89.93247
## Dim.19 0.532431274       1.02390630                 90.95637
## Dim.20 0.484840952       0.93238645                 91.88876
## Dim.21 0.425640834       0.81854007                 92.70730
## Dim.22 0.398595212       0.76652925                 93.47383
## Dim.23 0.382694691       0.73595133                 94.20978
## Dim.24 0.339300889       0.65250171                 94.86228
## Dim.25 0.307706521       0.59174331                 95.45402
## Dim.26 0.292964236       0.56339276                 96.01742
## Dim.27 0.255991614       0.49229157                 96.50971
## Dim.28 0.236252841       0.45433239                 96.96404
## Dim.29 0.203445985       0.39124228                 97.35528
## Dim.30 0.179889105       0.34594059                 97.70122
## Dim.31 0.170113804       0.32714193                 98.02837
## Dim.32 0.131742348       0.25335067                 98.28172
## Dim.33 0.121832106       0.23429251                 98.51601
## Dim.34 0.112447021       0.21624427                 98.73225
## Dim.35 0.091981456       0.17688742                 98.90914
## Dim.36 0.079718822       0.15330543                 99.06245
## Dim.37 0.064211387       0.12348344                 99.18593
## Dim.38 0.056537299       0.10872558                 99.29465
## Dim.39 0.055188020       0.10613081                 99.40079
## Dim.40 0.040801837       0.07846507                 99.47925
## Dim.41 0.038103474       0.07327591                 99.55253
## Dim.42 0.035457709       0.06818790                 99.62071
## Dim.43 0.033727650       0.06486087                 99.68557
## Dim.44 0.032215407       0.06195270                 99.74753
## Dim.45 0.028716975       0.05522495                 99.80275
## Dim.46 0.026853460       0.05164127                 99.85439
## Dim.47 0.021661899       0.04165750                 99.89605
## Dim.48 0.020595887       0.03960747                 99.93566
## Dim.49 0.013471638       0.02590700                 99.96157
## Dim.50 0.011875304       0.02283712                 99.98440
## Dim.51 0.005961537       0.01146449                 99.99587
## Dim.52 0.002148931       0.00413256                100.00000
```

```
fviz_eig(pcaCov)
```



Although the correlation between covariates is not big, a principal components analysis show that with only 10 components (from 52) it is possible to explain 90% of the total variation.

4. MODELING

All modeling will be done considering a parallel computation using **doParallel** package.

```
workers <- availableCores() - 1
```

4.1 Multinomial Regression

The first model that will be tested it is multinomial regression. Two models will be done: (1) all with centar an scale and with and without principal component as a pre-processing.

```
cl <- makeClusterPSOCK(workers)

registerDoParallel(cl)

mdlLrSc <- train(classe ~., data = train, method = 'multinom', preProcess = c("center","scale"))

stopCluster(cl)

registerDoSEQ()
```

```
lrScPred <- predict.train(mdlLrSc, newdata = test)

print(confusionMatrix(lrScPred, reference = as.factor(test$classe)))
```

Only Center and Scale

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 961  89  60  50  30
##          B  35 481  54  28  93
##          C  56  93 480  71  65
##          D  50  21  63 458  41
##          E  14  75  27  36 492
##
## Overall Statistics
##
##                Accuracy : 0.7321
##                  95% CI : (0.7179, 0.7459)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6605
##
##  Mcnemar's Test P-Value : 5.033e-09
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.8611   0.6337   0.7018   0.7123   0.6824
## Specificity            0.9184   0.9336   0.9120   0.9466   0.9525
## Pos Pred Value         0.8076   0.6961   0.6275   0.7235   0.7640
## Neg Pred Value         0.9433   0.9140   0.9354   0.9438   0.9302
## Prevalence             0.2845   0.1935   0.1744   0.1639   0.1838
## Detection Rate         0.2450   0.1226   0.1224   0.1167   0.1254
## Detection Prevalence   0.3033   0.1761   0.1950   0.1614   0.1642
## Balanced Accuracy      0.8898   0.7837   0.8069   0.8295   0.8175
```

The multinomial regression, centering and scalling the variables, was able to achieve a accuracy of 0,73 on test dataset.

```
cl <- makeClusterPSOCK(workers)

registerDoParallel(cl)

mdlLrPCA <- train(classe ~., data = train, method = 'multinom', preProcess = c("center", "scale", "pca")

stopCluster(cl)
```

```
registerDoSEQ()
```

```
lrPCAPred <- predict.train(mdlLrPCA, newdata = test)

print(confusionMatrix(lrPCAPred, reference = as.factor(test$classe)))
```

Center, Scale and PCA

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction   A   B   C   D   E
##          A 768 185 195  60  62
##          B  94 301  67  94 142
##          C  90 109 336  76  89
##          D 125  88  49 334  87
##          E  39  76  37  79 341
##
## Overall Statistics
##
##                Accuracy : 0.5302
##                  95% CI : (0.5144, 0.5459)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4031
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.6882  0.39657  0.49123  0.51944  0.47295
## Specificity            0.8212  0.87453  0.88762  0.89360  0.92786
## Pos Pred Value         0.6047  0.43123  0.48000  0.48902  0.59615
## Neg Pred Value         0.8688  0.85798  0.89203  0.90463  0.88660
## Prevalence             0.2845  0.19347  0.17436  0.16391  0.18379
## Detection Rate         0.1958  0.07673  0.08565  0.08514  0.08692
## Detection Prevalence   0.3237  0.17793  0.17843  0.17410  0.14581
## Balanced Accuracy      0.7547  0.63555  0.68942  0.70652  0.70041
```

Using PCA as a pre-processing have decreased the accuracy on test set.

4.2 Random Forest

To improve the prediction capabilities, a random forest will be used, considering all hyperparameters as defautl values.

```
cl <- makeClusterPSOCK(workers)

registerDoParallel(cl)

mdlRf <- train(classe ~., data = train, method = 'ranger')

stopCluster(cl)

registerDoSEQ()
```

```
rfPred <- predict.train(mdlRf, newdata = test)

print(confusionMatrix(mdlRf, reference = as.factor(test$classe)))
```

```
## Bootstrapped (25 reps) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction    A    B    C    D    E
##          A 28.1  0.2  0.0  0.0  0.0
##          B  0.1 19.2  0.2  0.0  0.0
##          C  0.0  0.1 17.2  0.2  0.0
##          D  0.0  0.0  0.1 16.1  0.0
##          E  0.0  0.0  0.0  0.0 18.3
##
##  Accuracy (average) : 0.9901
```

USing only the default values of hyperparameters, it was possible to increase the accuracy to 0.99. To have a second view of the performance, the model will be tested on validation data set.

```
rfval <- predict.train(mdlRf, newdata = val)

print(confusionMatrix(mdlRf, reference = as.factor(val$classe)))
```

```
## Bootstrapped (25 reps) Confusion Matrix
##
## (entries are percentual average cell counts across resamples)
##
##           Reference
## Prediction    A    B    C    D    E
##          A 28.1  0.2  0.0  0.0  0.0
##          B  0.1 19.2  0.2  0.0  0.0
##          C  0.0  0.1 17.2  0.2  0.0
##          D  0.0  0.0  0.1 16.1  0.0
##          E  0.0  0.0  0.0  0.0 18.3
##
##  Accuracy (average) : 0.9901
```

5. FINAL MODEL EVALUATION

Final prediction for 20 selected cases.

```r
finalPred <- predict.train(mdlRf, newdata = harTstClean)

finalPred
```

```
##  [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
```