Prediction Assignment Writeup

Stefano Emilio Campanini

February 27, 2016

Executive Summary - 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. The goal of this project is to create a model in order to predict the manner in which the exercise is done, starting from the *Weight Lifting Exercises Dataset* .

Environment and libs

The Environment, the required libs and optimizations used for parallel processing, other are reported in Appendix

The Data

The data used is available from the Groupware@LES (http://groupware.les.inf.puc-rio.br/har), and it is collected asking to 6 participants to perform one set of 10 repetitions of the Unilateral Dumbbell Biceps Curl in five different fashions. The training data set collect 19622 observations with 160 fields, the outcome is named "classe". The test data set collect 20 observations with 160 fields, the outcome is missed. The outcome of this dataset will be predicted by the model created and submitted for evaluation.

Two data set are created, as reported below, The code used to load these data sets is reported in Appendix.

- *trainingForFitting* , from the fle *pml-training.csv*
- *testingForResult* , from the fle *pml-testing.csv*

The way we apply the Cross Validation in this project is by using the *trainingForFitting* dataset to create the *training* and *testing* data set.

```
inTrain <- createDataPartition(y=trainingForFitting$classe, p=0.75, list=FALSE)
training <- trainingForFitting[inTrain, ]
testing <- trainingForFitting[-inTrain, ]
```

The *training* dataset is used to do the analysis and for fitting the model, instead the *testing* dataset is only used to evaluate the quality of the model (validation).

The *testingForResult* dataset is used to calculate predictions to be used to answer to the last quiz of the course.

Data Pre-Processing

Exploration, Cleaning, Selecting features, and Preprocessing

In the dataset there are more fields than necessary. Infact, fileds like *timestamps* , *user_name*, *window*, *program_id*, ecc.. that reasonably cannot influence the outcome. So we keep only some measurements produced by sensors. Below are reported the fields we selected as features for the model, and its dimensions.

```
fieldsToKeep <- c("roll_belt","pitch_belt","yaw_belt","total_accel_belt","gyros_belt_x","gyros_belt_y",

filteredTraining <- training[,  colnames(training) %in% fieldsToKeep]
dim(filteredTraining)
```

```
## [1] 14718    53
```

We do not check before how much these features are correlated, supposing some of them can be correlated each each others we decide to apply PCA preprocessing with .95 threshold for selecting components. This is also useful to reduce the number of predictors.

Also we decide to use K–fold cross–validation using R default K parts, and in order to speedup the model fitting algorithm the features will be "scaled" and "centered" .

The preprocessing for *K–fold*, *center*, *scale* and *pca* , is done during the model fitting using *trainControl* and *train* options, of the *caret* package.

Model Fitting

This is a machine learning supervised classification problem, we choose to use *Random Forest* with parallel processing algorithm, with a number of three lower than default. Here the code

```
set.seed(96)
ctrl <- trainControl(preProcOptions = list(thresh = 0.95) , method = "cv", allowParallel =TRUE)
modfit <- train(classe ~ .,
   preProcess=c("center", "scale", "pca") ,
   trControl=ctrl,
   data = filteredTraining,
   method = "rf", ntree = 200)
```

Model Quality

In appendix it is reported the model *confusion matrix* and the way errors converge.

Validation of the Model

In these section, it is reported the model validation against the testing data.

```
filteredTesting <- testing[,  colnames(testing) %in% fieldsToKeep]
predictions <- predict(modfit, newdata = filteredTesting)
cm <- confusionMatrix(predictions, filteredTesting$classe)
```

Here are reported *onfusion Matrix and Statistics* about computed using testing data, as you can see the overall accurancy is about 0.6 with a confidence interval from 0.5008 to 0.6971 .

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction    A    B    C    D    E
```

```
##          A 1389   13    0    1    0
##          B    1  916   16    0    4
##          C    3   17  820   26    6
##          D    1    0   18  774    5
##          E    1    3    1    3  886
##
## Overall Statistics
##
##                Accuracy : 0.9757
##                  95% CI : (0.971, 0.9799)
##     No Information Rate : 0.2845
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.9693
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: A Class: B Class: C Class: D Class: E
## Sensitivity            0.9957   0.9652   0.9591   0.9627   0.9834
## Specificity            0.9960   0.9947   0.9872   0.9941   0.9980
## Pos Pred Value         0.9900   0.9776   0.9404   0.9699   0.9911
## Neg Pred Value         0.9983   0.9917   0.9913   0.9927   0.9963
## Prevalence             0.2845   0.1935   0.1743   0.1639   0.1837
## Detection Rate         0.2832   0.1868   0.1672   0.1578   0.1807
## Detection Prevalence   0.2861   0.1911   0.1778   0.1627   0.1823
## Balanced Accuracy      0.9959   0.9800   0.9731   0.9784   0.9907
```

Results for submission

We export the results for the submission in a CSV format, the same format of the input data sets. This data will be used to answer to the last quiz of the course. In Appendix is reported the code used.

Appendix

Enviroment and libs

This analysis has been made using R programming language and RStudio IDE, below are reported useful details about the base software environment.

The Hardware is:

- memory size: 15GiB
- cpu product: Intel(R) Core(TM) i7-3612QM CPU @ 2.10GHz

Interesting software

```
sessionInfo()
```

```
## R version 3.2.2 (2015-08-14)
```

```
## Platform: x86_64-pc-linux-gnu (64-bit)
## Running under: Ubuntu 15.10
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=it_IT.UTF-8        LC_COLLATE=en_US.UTF-8
##  [5] LC_MONETARY=it_IT.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=it_IT.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=it_IT.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
## [1] parallel  stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
## [1] doParallel_1.0.10   iterators_1.0.8     foreach_1.4.3
## [4] randomForest_4.6-12 ipred_0.9-5         caret_6.0-64
## [7] ggplot2_2.0.0       lattice_0.20-33
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.3         compiler_3.2.2      formatR_1.2.1
##  [4] nloptr_1.0.4        plyr_1.8.3          class_7.3-13
##  [7] tools_3.2.2         rpart_4.1-10        digest_0.6.8
## [10] lme4_1.1-10         evaluate_0.8        nlme_3.1-122
## [13] gtable_0.1.2        mgcv_1.8-7          Matrix_1.2-2
## [16] yaml_2.1.13         prodlim_1.5.7       SparseM_1.7
## [19] e1071_1.6-7         stringr_1.0.0       knitr_1.11
## [22] MatrixModels_0.4-1 stats4_3.2.2        grid_3.2.2
## [25] nnet_7.3-10         survival_2.38-3     rmarkdown_0.9
## [28] lava_1.4.1          minqa_1.2.4         reshape2_1.4.1
## [31] car_2.1-1           magrittr_1.5        scales_0.3.0
## [34] codetools_0.2-14    htmltools_0.2.6     MASS_7.3-45
## [37] splines_3.2.2       pbkrtest_0.4-5      colorspace_1.2-6
## [40] quantreg_5.19       stringi_1.0-1       munsell_0.4.2
```

Libs used

```
library(caret)
library(ggplot2)
library(ipred)
library(randomForest)
```

Performance optimization - Tweaking for parallel processing

The model will use a high number of covariate/predicotrs, so fitting it can be a CPU intensive and long task. it is better to configure R using parallel processing. Here the code used

```
library(doParallel);
rCluster <- makePSOCKcluster(detectCores());
registerDoParallel(rCluster);
```

Loading the Data

The data is lodaed using read.csv, as you can see there is the need to force the field type for the loading of the test dataset. The read.csv produce fields as *logical* if there are no data, here we forced the types using colClasses in order to assure that the two datasets will be aligned by field-types.

```
trainingForFitting <- read.csv("pml-training.csv")
testingForResult <- read.csv("pml-testing.csv", colClasses = c(kurtosis_roll_belt="factor", kurtosis_pi
```

Model Quality

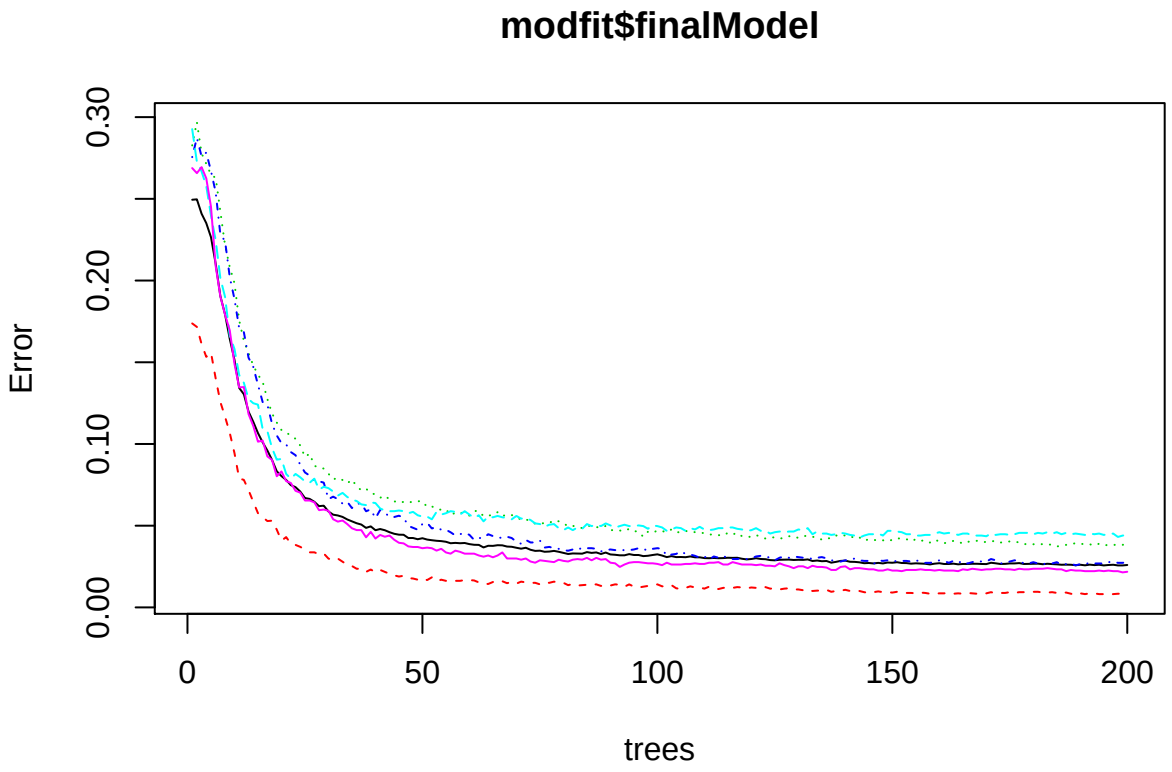
Here, it is reported the error rates of the model and the confusion matrix.

```
modfit$finalModel
```

```
##
## Call:
##  randomForest(x = x, y = y, ntree = 200, mtry = param$mtry)
##                Type of random forest: classification
##                      Number of trees: 200
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 2.6%
## Confusion matrix:
##      A    B    C    D    E class.error
## A 4151   11   10   10    3 0.008124253
## B   51 2737   55    3    2 0.038974719
## C    6   34 2496   24    7 0.027658746
## D    6    0   90 2305   11 0.044361526
## E    0   21   20   18 2647 0.021803400
```

Thre plot shows how the erorrs change respect the number of trees, as you can notice it converged also with less than 500 trees (default value).

```
plot(modfit$finalModel)
```



Results for submission

We export the results for the submission in a CSV format, the same format of the input data sets. Here the code used. This data will be used to answer to the last quiz of the course.

```
filteredTestingForResult <- testingForResult[,  colnames(testingForResult) %in% fieldsToKeep]
predictions4Eval <- predict(modfit, newdata = filteredTestingForResult)
predictions4EvalDF <- data.frame(predictions4Eval)
predictions4EvalDF$classe <- predictions4EvalDF$predictions4Eval
predictions4EvalDF$predictions4Eval <- NULL
write.csv(x=predictions4EvalDF, "test-result-4-eval.csv", row.names=FALSE)
```