Machine Learning Writeup

Eduardo B. Díez — August 2014

Introduction

This document is born with the vocation to be accessible as a web page from this gh-page or a copy in rpubs and the source code —as a Rmd file— and the compile result —as a HTML file— can be found in this github repository. The report try to explain how we chose the forecast for 20 cases about the way in which six participants do exercises according with the *datae* from this link to finally submit the forecast as the final part of the project.

We'll talk of the predictors that were selected as "features" to develop the model and about the various training methods we used whose results were compared and provided aid to make the final decision based on the accuracy they showed.

Bunch of Data and very Few Features

In this project we will use data from four accelerometers on the belt, forearm, arm, and dumbbell of six participants. They were asked to perform barbell lifts correctly and incorrectly in 5 different ways. So, the data contain the raw information from the gyroscopes and also some variables derivates from the raw as mean, standard deviation among other few.

After the inspection and work around with the data we arrive to the conclusion that the variables we can understand as real features with added value to be used to model the problem effectively are those about roll, yaw, pitch and total acceleration that correspond with each of the four locations where the accelerometers are registering the movements.

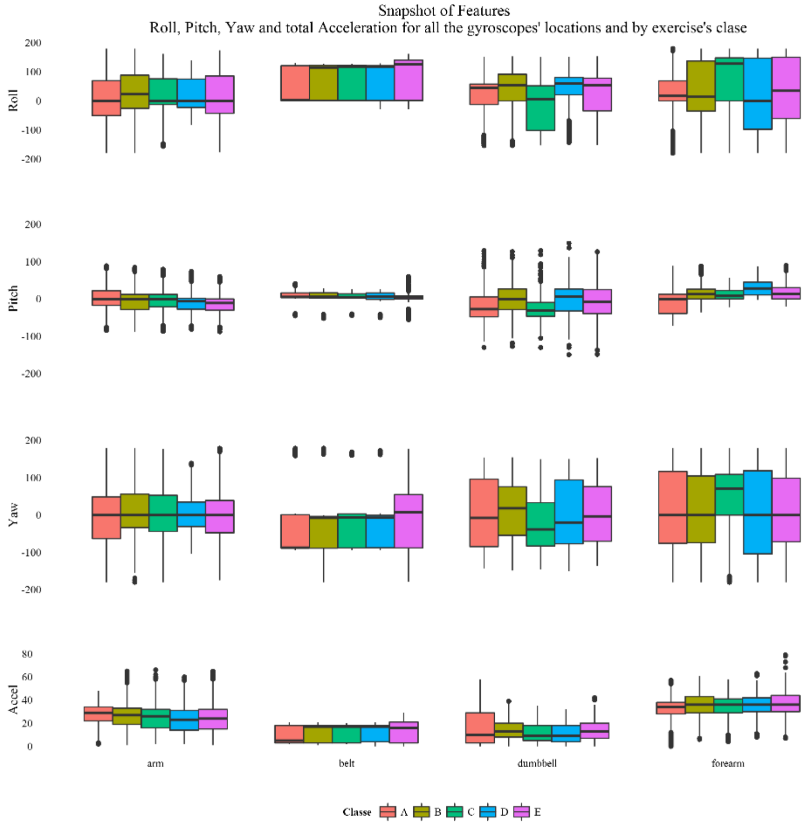

Figure 1. Initial set of features selected; Roll, Pitch, Yaw and total acceleration by the localization of each accelerometer. Also, each class of exercise is associated by a color.

After removal the other variables, we proceeded to the proper ordering of the remaining to present them graphically, so one can easily detect any anomaly that may exist and whereafter was computed the correlation between them. The variable **roll_bell** was found correlated with other two of them thus, we decided to eliminate it from the data set.

Below we show the 15 variables with which we make our model.

#	Feature
1	pitch_belt
2	yaw_belt
3	total_accel_belt
4	roll_arm
5	pitch_arm
6	yaw_arm
7	total_accel_arm
8	roll_dumbbell
9	pitch_dumbbell
10	yaw_dumbbell
11	total_accel_dumbbell
12	roll_forearm
13	pitch_forearm
14	yaw_forearm
15	total_accel_forearm

Summary of predictors to use.

Machineries

Before proceeding is important to note the following to the reader: In the source document final.Rmd the chunk corresponding to this point has the option eval = false to avoid running this code which is very intensive in CPU load and run time. Additionally, this chunk with name “Machineries” has several other models that have not been included in the writing of this paper —plots and tables— for lack of space, but is operational and again time consuming.

Once the data was prepared, the data set was partitioned 60-40 between training and testing. All methods have similar schemes of cross-validation control with the default values, except for the two random forest, RF1 and RF2, which employ their own bootstrap characteristics in order to evaluate the OOB. Also, one boosting method was include, the GBM. In the source code can be found lots of additional calculations to check the results, prior to show and resume them.

Follow a panel figure of ROC plots that provide tools to select possibly optimal models and to discard sub-optimal ones independently from the class. In this panel, we can observe the good response of the KNN method —k-nearest neighbor algorithm— beside the two random forest ones. Every plot shows, additionally, the process time inverted for each particular algorithm and the whole figure give us an idea of the extensive variety or results.

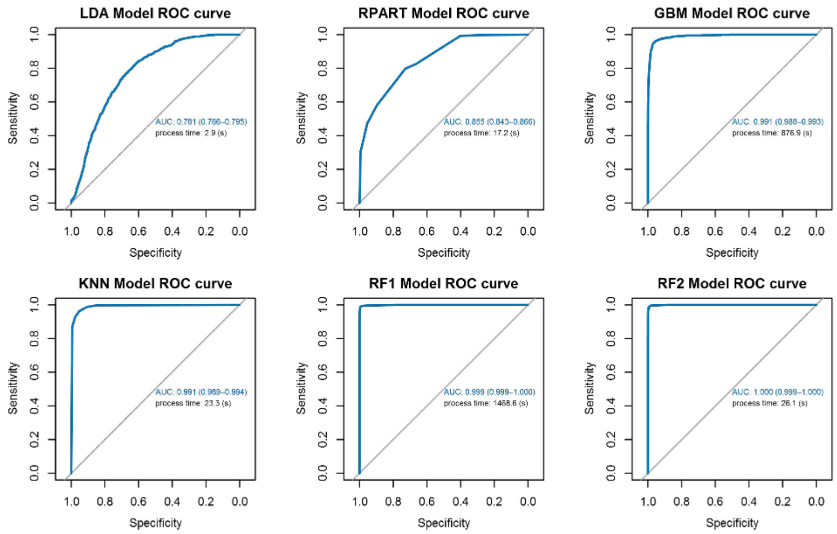

Figure 2. The ROC curves for the models with the total AUC (Area Under the Curve) with 95% confidence interval and the total process time running in Mobile Intel Core 2 T6400 at 2GHz with 4GB

Results

We're going to use the results of *confusionMatrix(…)$**byClass*** to construct tables from each method that we used, in order to analyze the statistical composing each matrix. For convenience in this report we show only two of them. The first corresponds to the result RF2 by using the the *randomForest()* function —with the best fit we got.

Table 1. Confussion Matrix Statistics of Dual RF2[†] model

randomForest()	**Class: A**	**Class: B**	**Class: C**	**Class: D**	**Class: E**
Sensitivity	0.99	0.99	0.96	0.98	1.00
Specificity	1.00	0.99	1.00	1.00	1.00
Pos Pred Value	0.99	0.97	0.98	0.99	0.99
Neg Pred Value	1.00	1.00	0.99	1.00	1.00
Prevalence	0.28	0.19	0.18	0.16	0.18
Detection Rate	0.28	0.19	0.17	0.16	0.18
Detection Prevalence	0.28	0.19	0.17	0.16	0.18
Balanced Accuracy	1.00	0.99	0.98	0.99	1.00

[†] The predictions of this model achieved a score of 20 / 20 in the submission part of the project,
same as RF1 and KNN models.

The second one correspond with the boosting GBM method by using the function *train(…, method = "gbm")*

Table 2. Confussion Matrix Statistics of Boosting GBM[†] model

method='gbm'	**Class: A**	**Class: B**	**Class: C**	**Class: D**	**Class: E**
Sensitivity	0.97	0.88	0.86	0.92	0.95
Specificity	0.98	0.97	0.98	0.99	0.99
Pos Pred Value	0.96	0.86	0.90	0.94	0.93
Neg Pred Value	0.99	0.97	0.97	0.98	0.99
Prevalence	0.28	0.19	0.18	0.17	0.18
Detection Rate	0.27	0.17	0.16	0.15	0.17
Detection Prevalence	0.28	0.19	0.17	0.16	0.18
Balanced Accuracy	0.97	0.93	0.92	0.95	0.97

[†] The predictions of this model achieved a score of 19 / 20 in the submission part of the project.

Final Submission

Now, we are going to use the results of the confusion matrix by *confusionMatrix(…)$**overall*** to construct a resume table with the values of *kappa* and accuracy —with its correspondent 95% ci— for each algorithm we've shown in this report in order to have the completed panorama of our finds.

Table 3. Summary of *Kappas* & Accuracies

			Accuracy CI		**Submission**
method	**Kappa**	**Accuracy**	**L: 2.5%**	**U: 97.5%**	**Score**
LDA	0.3088	0.4560	0.4450	0.4671	11/20
RPART	0.3992	0.5213	0.5102	0.5324	4/20[†]
KNN	0.8977	0.9191	0.9128	0.9250	20/20[‡]
GBM	0.9012	0.9219	0.9157	0.9277	19/20[†]
RF1	0.9776	0.9823	0.9791	0.9851	20/20[‡]
RF2	0.9803	0.9845	0.9815	0.9871	20/20[‡]

[†] RPART and GBM methods show that a model which fits the data well does not necessarily forecast well. We should expect RPART doing it better than LDA and GBM than KNN but, it is not the case.
[‡] KNN, RF1 and RF2 agree with the same 20 responses to be submited.

We observe easily, by inspecting Table 3, that the two best accuracy results, both about 98%, agree in the 20 answers —prior to make the submission— and there is a third algorithm that also matches these responses. So, with a 98% accuracy and commonality between three different algorithms, all with high accuracy, cause us to be confident that these are the answers —the predictions of those three methods— we will submit.

And so we did and got a correct result in all 20 answers.

Additionally, the table has added the submission results virtually achieved for each method, once we submitted the answers and got a score of 20 / 20 in the first try we did.

About this Document

All analyses were performed using *R version 3.1.1 (2014-07-10)* and *RStudio* as IDE, with the default base packages *parallel, splines, grid, stats, graphics, grDevices, utils, datasets, methods, base*, additionally *BradleyTerry2, brglm, car, class, cluster, codetools, coin, colorspace, digest, evaluate, foreach, formatR, gtable, gtools, htmltools, iterators, kernlab, labeling, latticeExtra, lme4, Matrix, mboost, minqa, modeltools, munsell, mvtnorm, nlme, nloptr, nnet, nnls, party, proto, quadprog, RColorBrewer, Rcpp, rmarkdown, RSNNS, sandwich, scales, sp, stats4, stringr, strucchange, tools, XML, yaml, zoo* and to produce this report in HTML the packages *Grmd* and *knitr*.