Rythm Machine Learning Challenge

Clément Tiennot – Mai 2016

Disclaimers

Data were provided by Rythm and all code provided in the Github repository is my own work. Otherwise specified, all accuracies are computed on the same test set which accounts for 20% of the data provided in the quality_dataset.h5 file.

Most of this report is about model selection and models performances but also feature engineering and understanding of the challenge overall. Some other approaches would have been possible, and I strongly believe that a deep learning oriented one could lead to great results, at the cost of extra-time and more requiring model parameters tuning.

I suspect that labels in the quality_dataset.h5 file are not in agreement with the description provided: it seems that 0 stands for good quality and 1 stands for bad quality. In the following I will treat them as such.

About the data

Data were retrieved from the Dreem headband and represent four EEG channels. We are interested in the quality of the signal which can be noisy during some period of the night. Specifically, the training dataset provided (quality_dataset.h5) focuses on windows of 2 second of EEG 250 Hz signal leading to 500 raw and 500 filtered measurements i.e. 1000 features per window. The dataset includes 137 030 observations/windows with their labels, that is 76 hours of sleep overall.

To assess our models performance, we will proceed in a "standard" way and split our data in a training and testing set with respectively 80% and 20% of the observations. Since the two classes distributions are approximatively balanced (45% of good quality) we do not really need to stratify the sampling scheme when splitting the data in train/test sets. Also this suggest the accuracy is a good metric to benchmark our algorithms. We could have considered more complex metrics such as the Area Under the (ROC) Curve or the logarithmic loss but without more background knowledge about the usage of the final models the accuracy seems to be good enough.

A simple benchmark

As a first model and a way to get insights about the challenge I trained a simple Random Forest classifier, first on a random subset of the data and then using the entire dataset. Provided enough trees are fitted (I chose 100 finally) Random Forest (RF) models are quite nice since they don't usually require much parameters tuning which is a great advantage especially when working under deadline as it was the case.

When using one tenth of the data the RF turns out to give a 91.8% accuracy and a 0.18 log-loss. Hence the two classes seem well separated given the data. Using the entire training dataset, the accuracy rises to 94.6% while the log-loss decreases to 0.18. Also the classifier took about 11 minutes to fit on my laptop (i5-3337U CPU 1.80 GHz).

Feature importances plot (mean decrease impurity) suggests that most useful splits use measurements from the filtered signal even if some measurements from the raw signal come out of the ground.

Exploratory Analysis

We could have come through an exploratory analysis in the first place, but benchmarking the problem will help keeping track of our models improvements in what follows.

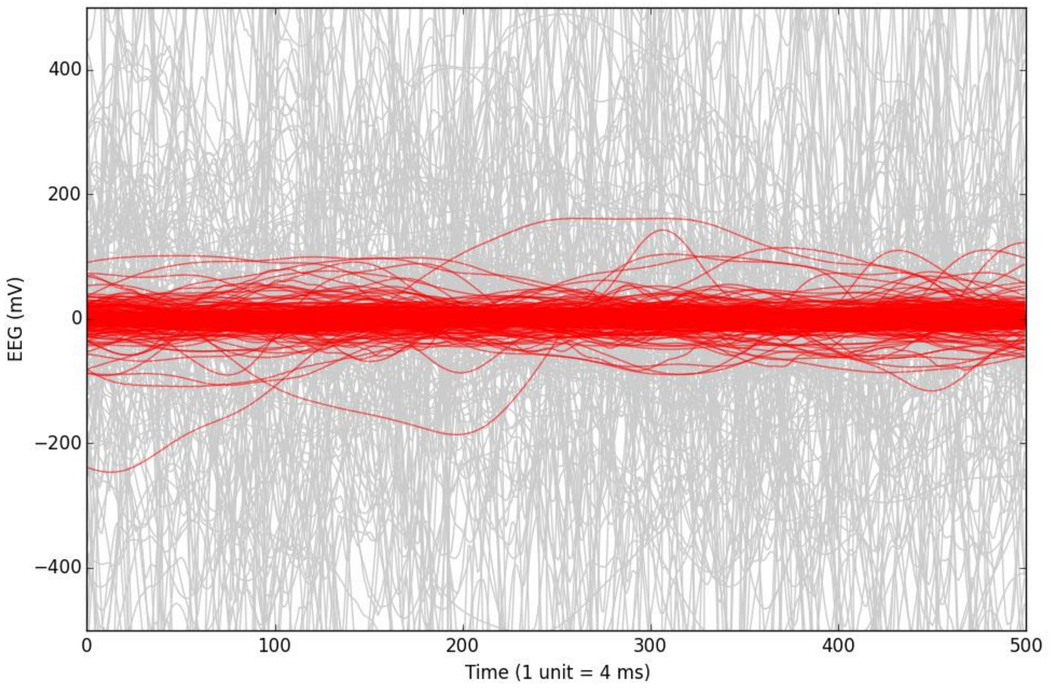

Figure 1: Random trajectories (red ones are good quality signals)

First we plot some trajectories from our filtered signals (Figure 1). We sample 300 bad quality and 300 good quality trajectories. Overall we understand why our classification classes are well separated: good signals are quite straight and do not exhibit much variation while bad quality signals (in grey) look rather noisy (the y-axis limits were set to ±500).

Figure 2 shows the variances distributions after log-transformation for both filtered signals and their increments, still stratified by labels. Let x_1, x_2, …, x_{500} be the measurements for one 2 seconds window, the increments are given by x_1- x_2, x_2- x_3, …, x_{500} – x_{499}. As suggested before, the variances appear as a good feature to differentiate between good and bad signal quality. Moreover, taking the increments instead of the signal itself seems to enhance the separation in the distribution and makes the variances computations more robust to changes in measurements scales. Hence from this point we will work with the increments.

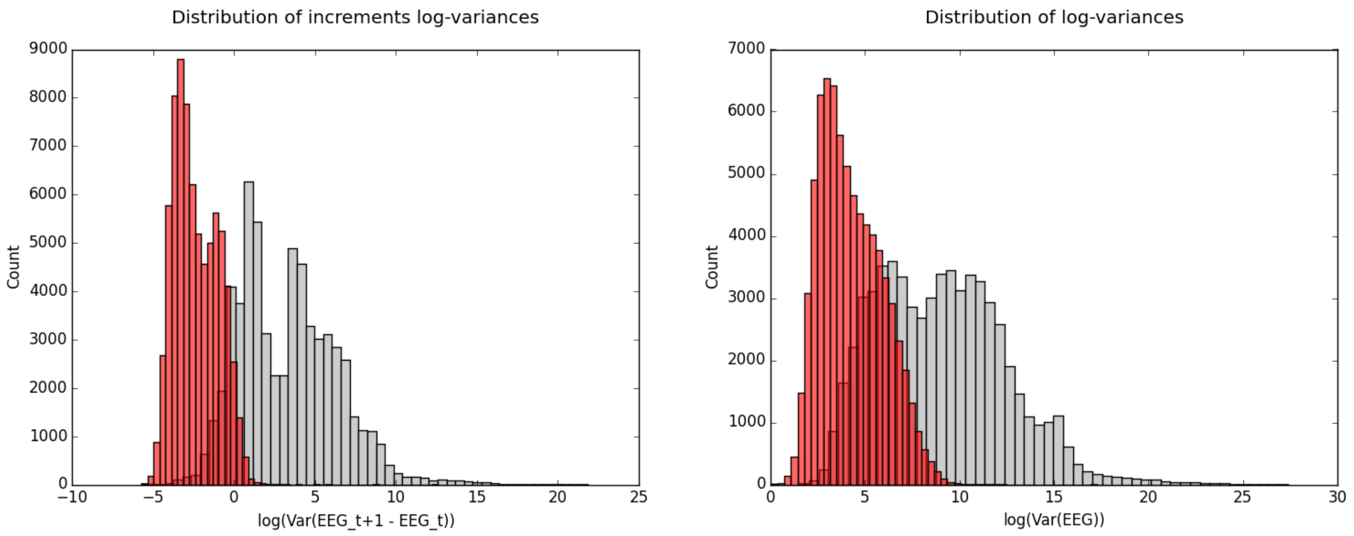

Figure 2: Log-variances distributions (red is good quality)

T

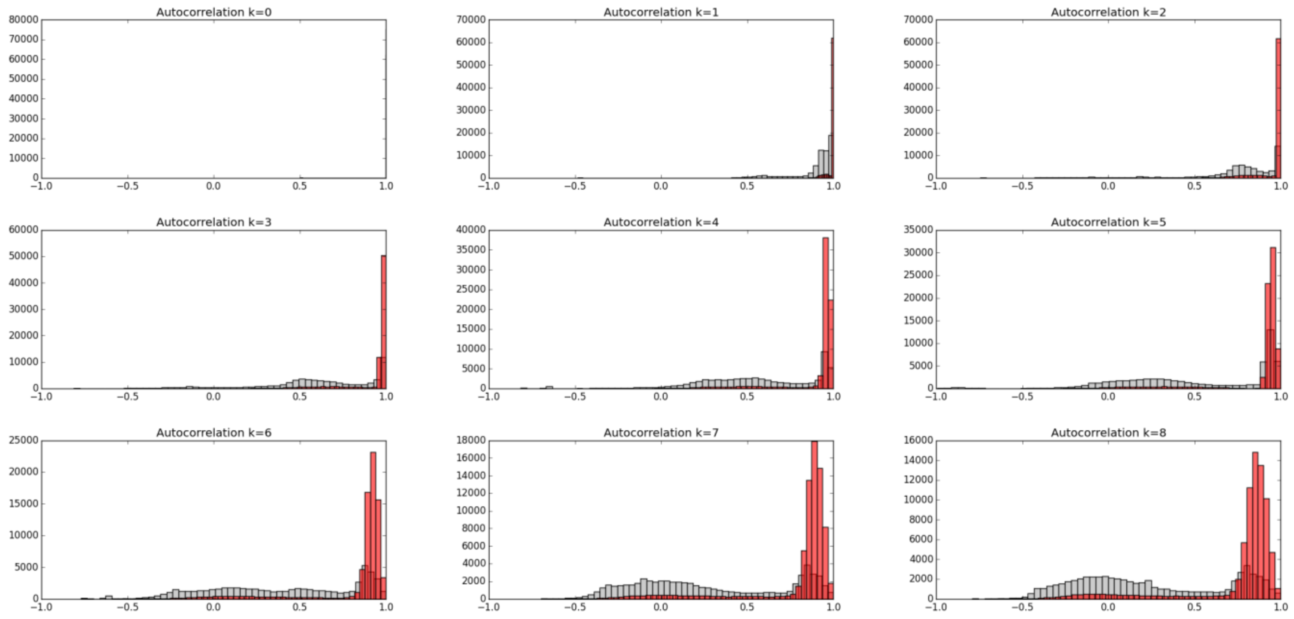

Figure 3: Filtered increments ACF (Auto Correlation Functions)

Feature engineering

Previous investigations strongly suggest that discriminating between good and bad signal quality is mainly about variability. A first natural idea would be to use the variance of each trajectory increments to classify using a threshold, which accounts for drawing a vertical line on Figure 2 between red and grey distributions. However, this classifier seems a bit simple and can not detect subtler behaviours like variability changes across time or even more complex interactions.

To extend the use of the global variance, we can separate each increments trajectory in many segments (a window length of 10 measurements proved good enough) and compute variances for each segments. We then end up with 50 variances for the filtered signal and as many for the raw signal.

The “final” model using a total of 102 features (50 piece-wise variances for the raw and the filtered signal plus the two corresponding overall variances) gives a 96.8 % accuracy and a 0.12 log-loss. It took about 3 minutes to fit and appears as a good performance/simplicity trade-off.

I also considered using the ACF (Auto Correlation Functions, see Figure 3) to account for the dependence between segments. Fitting a Random Forest model on top of the piece-wise variances plus the ACF (until 9th order) led to a small improvement (97.5% accuracy, 0.10 log-loss), potentially non-significant. For the sake of simplicity and also because the ACF require a bit of extra computations I preferred not to use them and kept only the piece-wise variances and the overall variances.

Interpretations

Overall the classification task is rather “easy” in the sense that a quite good accuracy can be achieved without so much effort. The extra artefacts used to increase the performance such as computing piece-wise variances or ACF are interesting and it would be wise to check whether the improvement is statistically significant.

One important thing to notice is that our models are using both the raw and the filtered signal. This was important in order to achieve the best accuracies. However, all feature importances outputs clearly show that models are primarily using the filtered signal which makes sense. The use of the raw signal seems moderate. Hence if computational efficiency is of interest, we could speed up the algorithm by fitting models on the filtered signal only or by increasing the segments lengths when computing variances in order to end up with less features, at the cost of a small decrease in accuracy.

New data prediction

To estimate the quality of 'record1.h5' and 'record2.h5' at each instant using our model, we broke each record in windows of 2 seconds (i.e. 500 measurements). We can then use our model(s) to predict the quality of each window/segment. In my implementation I chose not to make the windows overlap each other, which actually makes our algorithm predict segments. Of course a real implementation could use a slipping window to get a more accurate estimate for each points or use overlapping windows and then retrieve the mean prediction. I didn’t go through it, but it is quite possible that it would raise some computation/memory allocation issues.

Confusion matrix - Channel 0

	Detected good	Detected bad
Good	0.24 %	0.00%
Bad	3.50 %	96.26 %

Confusion matrix - Channel 1

	Detected good	Detected bad
Good	77.83 %	1.83 %
Bad	1.57 %	18.76 %

Figure 4 shows the estimates for the 4 channels of 'record1.h5' dataset. Red parts highlight detection of poor signals. Since some labels were provided for this dataset, it is interesting to check the accuracy and the confusion matrices obtained. For channel 0 and channel 1 (no labels were provided for the other channels), the accuracies are close to the numbers provided before: 96.5 % and 96.50 % for channel 0 and channel 1 respectively. Interestingly, if the accuracies are similar, the confusion matrices reveal two completely different patterns: barely no “Good” labels are present in channel 0 leading to highly imbalance classes while around 80% of the labels are “Good” for channel 1.

The imbalanced nature of the classes can be either due to a bias in the way data were classified (e.g. by segments) or to the signals themselves. Luckily enough, our classifiers still predict “well” but this could be subject to further investigation by for instance considering outputting probabilities instead of plain predictions, before choosing a threshold according to our need.

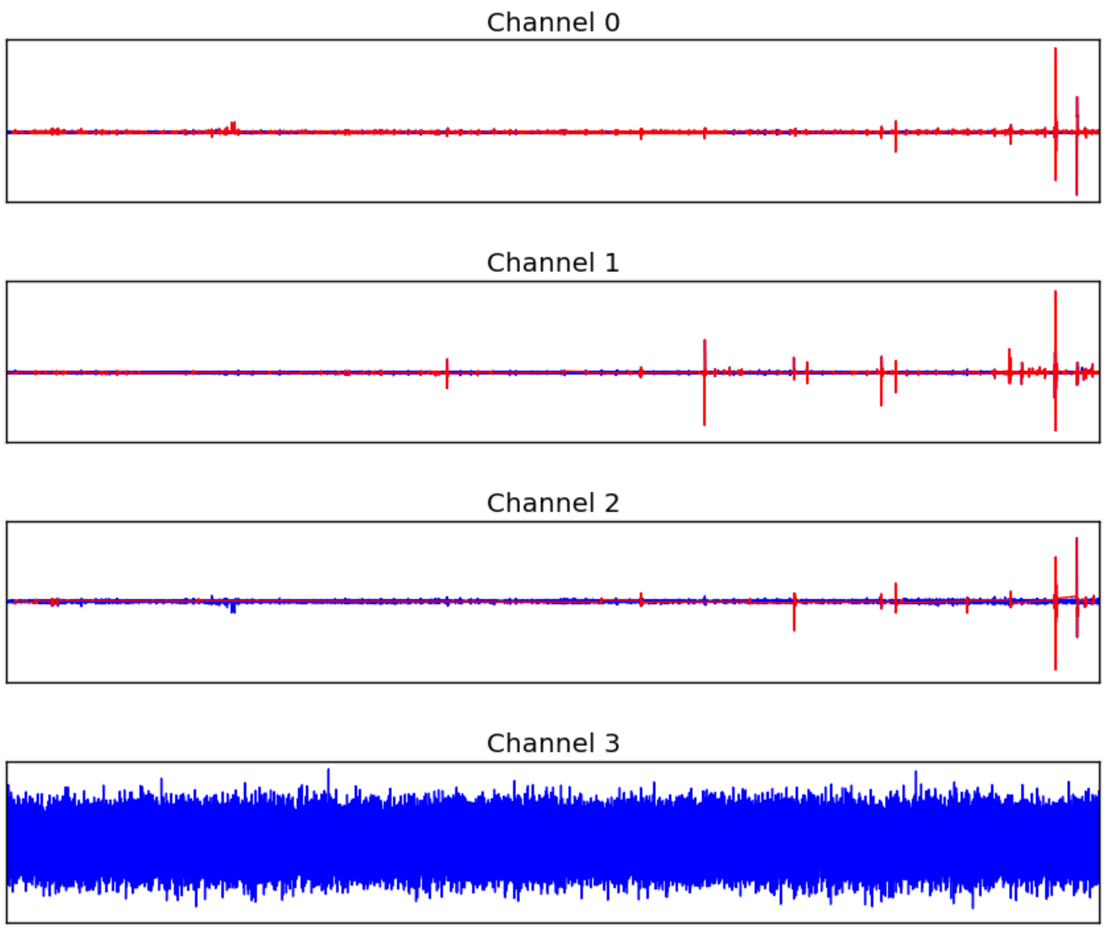

Figure 4: estimates for record 1 data (red parts are labelled as bad quality)

Conclusion

Overall we built a working quality detector, based on Random Forests and feature extraction. Other algorithms such as Gradient Boosting have been considered on the go, but no improvement compared to the Random Forest were found. Of course, tuning parameters and considering other approaches could lead to more precise models but this is beyond the scope of this challenge.