Development of Signal Quality Indices (SQIs) for physiological data

Célia Benquet, *MSc student*,
celia.benquet@gmail.com
Fall 2021

***Abstract*—Continuous monitoring is a key aspect to improve the patient's journey going through a medical procedure. An obvious utility being monitoring the recovery period after a surgery, if the monitoring device is practical enough, it also has a high number of other applications. It answers to the current problematic linked to hospital overload or patient comfort and personalized care. However, in order to be fully integrated to this journey, the monitoring device must be efficient in capturing physiological or technical anomalies, corresponding to respectively complications or artefacts, while providing enough data for the practitioner to understand clearly the state of the patient. This project aims at improving the detection of technical anomalies and noises in ECG on one hand and PPG on the other hand. Regarding ECG signals, different machine learning models are evaluated on different datasets. Kernel Density Estimation seems to be a good trade-off between availability and performances while being easily implementable in an online process as it can be trained with clean data only. For PPG signals, a method using covariance matrix and Riemannian geometry is investigated. Results are encouraging as it performs as good as the existing SQIs for clean and burn-in data. It also seems to be capturing shifts between the PPG signals, which is not done with the current algorithm.**

I. Introduction

RDS (Rhythm Diagnostic Systems) aims at developing the MultiSense® strip, the first miniaturized and connected wearable medical strip for real-time monitoring of several key cardiac and respiratory parameters. We seek at having the patch used for post-surgery check-up, which would ultimately allow the patients to go back home sooner while still being monitored, improving the patient's confort while liberating space in crowded hospitals. But recording real-time data in the everyday life of a subject necessarily makes signal quality one of the main challenges for the Multisense® patch. Hence, being able to efficiently discriminate between clean and noisy signals is of great interest to add value to our product. However, as we still aspire to have enough raw signal to derive information from it, the question of availability of data also needs to be raised and a trade-off between performances of the classifier and resulting availability will be investigated.

In the following paper, we investigate Signal Quality Indexes (SQI) for both ECG (electrocardiogram) and PPG (photoplethysmogram) signals. SQI is defined as a value computed from a signal to qualify its quality. This index can be calculated from different methods, using machine learning tools or more simple data separation techniques. Two different investigations, for respectively ECG and PPG are performed.

Previous literature has proposed diverse discriminating algorithms for ECG. Most of them combine the extracted features to directly get their SQI. Zhao et al. (2018) combine 6 feature, namely R-peak detection matching degree, QRS wave power spectrum distribution, variability of the R-R interval, skewness and kurtosis, and baseline relative power, to generate their SQI and use them to discriminate between good and bad signals [1]. Orphanidou et al. uses the correlation coefficiant between the average QRS template and each individual QRS complex as their SQI [2]. Currently, the RDS algorithms qualify the quality of the signal based on univariate models with a threshold-based classification by combining different SQIs to adapt to each type of noise.

However, those methods are limited in the number of noises they can discriminate. We are interested in developing an SQI that could be trained continuously with new data, plus that would be robust to more types of noises. For that, similarly to state-of-the-art literature, we extract features of interest from the individual signals but instead of combining them in a simple manner, we use them as input data to supervised learning and density estimation models. We investigate more precisely machine learning methods that could learn a *normality* distribution so that all noises are caught using the same SQI.

Characteristic clean ECG signal is presented on Fig. 6.a. We identify 7 different types of noises that we would like to be able to discriminate, presented in Table I with sample examples on Fig. 6 (Appendix). EMG (electromyography) and motion noises are observable when the patient is active. Step noise corresponds to a sudden movement. Low R and lead off noises indicate wrongly placed device, respectively the device itself and the electrocardiogram lead. All those noises are visible only looking at the ECG signal. Hence, investigated methods consist in learning a normality model based on features extracted from the ECG signal, that can reflect differences between clean signal and those types of

anomalies. To test those techniques, we use different kind of datasets, based on real clinical data and artificial data.

ECG	PPG
clean	clean
lead off	shift
EMG	venous
step	patch_off
low R	movements
motion	spike
powerline	respiratory

Table I: *Type of noises that can be foudn in ECG and PPG signals respectively.*

PPG signals are used to detect blood volume changes in the microvascular bed of tissue by measuring light reflection. The MultiSense® solution records PPG signals for two different channels: red and infra-red. From those signals, our algorithms are able to derive information such as blood oxygenation rate. Anomalies that can happen in PPG are presented in Table I. Similarly to ECG, PPG anomalies modify the signal such that accuracy of the derived information is compromised. However, if they are, in part, similar to the ones in ECG signals, some anomalies are due to phase shifts or venous modulation of the PPG channels with one another or with the ECG signal (see Fig. 1 for an example). Those can only be detected by comparing multiple signals. Currently, our algorithms cannot detect those shifts. Considering that the anomalies are related to how much the different signals change together, we hypothesize that looking at the correlation of the signals and learning how signal should *normally* evolve could be enough to detect artefacts.

And indeed, Barachant et al. (2013) present the Riemannian Potato model, an artifact detection method for online experiment, validated on EEG (electroencephalogram) channels [3]. The Riemannian Potato uses covariance matrices as descriptors of the signals and a Riemannian metric to compare these covariance matrices with an average covariance matrix estimated on the signal baseline.

II. Methods

A. SQIs for ECG

1) Features: In order to represent the signals, some features that capture the characteristics of the signals need to be extracted. The same features are extracted for all databases (see details on databases in Section II-A2): *mean, standard deviation, skewness, power spectral density for different ranges, between 1Hz and 3Hz, below 20Hz and above 20Hz, powerline ratio and cepstral distance to white noise*. Those features are calculated directly on the epochs from the database. All features are log-transformed. Details are provided for features that we used directly as SQI scores (see Section II-A3).

a) Standard Deviation: The standard deviation gives information on variation of the R-peak amplitude. The current discrimination algorithm at RDS for ECG is based on the standard deviation of the signal plus the standard deviation of the R-R interval.

b) Powerline ratio: The powerline ratio provides information on the frequency rate that is potentially corresponding to powerline in the signal by looking at the ratio of frequencies around 50Hz (powerline in Europe) and 70Hz (powerline in US) compared to the full range of available frequencies. This feature is currently used, in combination to the standard deviation in the RDS algorithm, for its performances in detecting lead-off (when electrode is disconnected from subject). Data is normalized in order to calculate this feature.

c) Cepstral distance to white noise: The cepstrum is the result of applying the inverse Fourier transform (IFT) of the logarithm of the signal power spectrum. It provides information on the periodicity of the signal. Hence, computing the cepstral distance between two signals gives a measure of their similarity. By computing similarities to white noise, we get a measure of quality of the signal under examination. Castiglioni et al. (2011) actually use cepstral distance directly as their SQI to assess ECG signal quality in real-time [4].

2) Datasets: A big part of developing and evaluating SQIs is having a solid dataset to try them on. To that extend, we designed 3 databases. A first database, that we will refer to as the *RDS dataset*, consists of real, hand-labelled RDS clinical data, containing all the type of signals recorded by the MultiSense® solution, meaning ECG, PPG, piezo, acceleration, temperature. The second dataset, namely *Neurokit dataset*, consists in artificial ECG data on which artificial noises, representing the different noises that can be found on real ECG data, were added. Finally, the last dataset, that we will call *MIT dataset*, uses the MIT-BIH Arrhythmia Database [5] on which the same artificial noises are applied randomly. All databases were designed so that the epoch size can be changed but the analysis was performed on 10-sec epochs (length of the epochs on which derivation is performed in the existing RDS algorithm).

a) RDS database: RDS database consists of manually-labelled studies recovered from the MultiSense® patch from volunteer subjects. Available signals are ECG, PPG, piezo, acceleration and temperature but we only use ECG. By looking at the full studies, the annotators have taken out sequences corresponding to a signal type of interest (see Table I) by reporting the start and stop offsets.

Each sequences consists in a given number of frames, with each frame corresponding to 1 second of measurements and so, knowing the sampling rates are 256Hz for ECG measurements. The annotators report the start and stop offsets of the sequences so that the corresponding raw data are loaded. Then, each labelled sequences are cut into epochs of fixed user-chosen size and features for each epoch are derived (see section II-A1). Only clean and lead-off data were annotated.

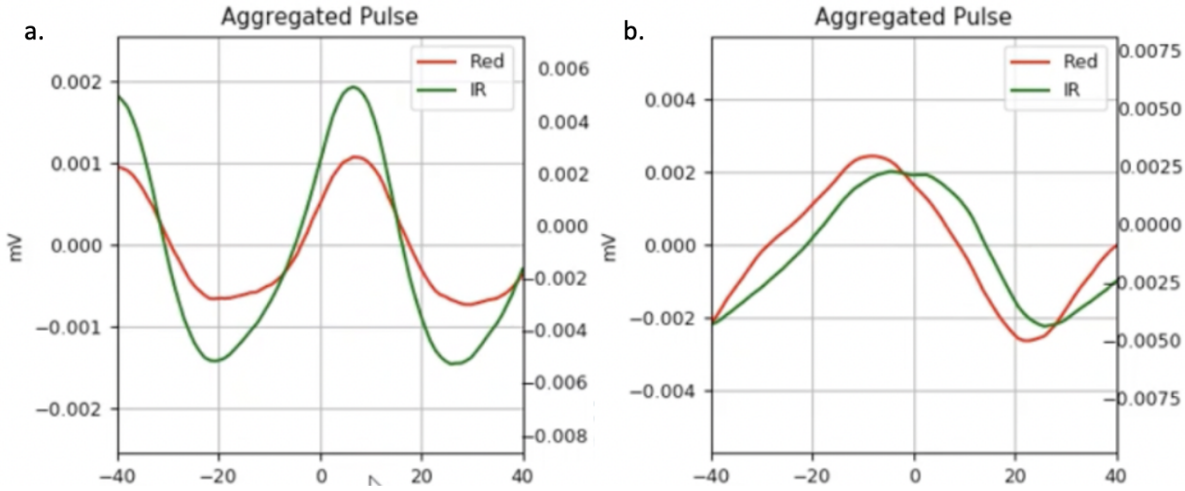

Figure 1: *Example of* ***a.*** *clean PPG signals and* ***b.*** *noisy PPG signals with a shift on the red channel.*

b) Neurokit database: This dataset consists in artificially built ECG on which different artificial noises, corresponding to noises observed in real clinical data, are added. The `neurokit2` library is used to generate clean ECG signals with varying heart rate and a Laplacian noise (similar to white noise) of amplitude of 0.05 (see Figure 6.a., Appendix, for a clean signal at 80 bpm) [6]. Then artificially generated noises can be randomly added on the clean signals (see Figure 6.b-g, Appendix). To label a sample, we use the heart rate (verified algorithm used for the MultiSense® device) as an information on its quality. We compute the true heart rate on the original sequence (that we know to be clean). Then, we compare it to the heart rate derived from the modified noisy sequence. It allows us to discriminate between weakly noisy signals that can be labelled as clean, because they can still be derived properly, and anomalies or highly noisy signals which need to be classified as noisy. We use the same heart rate difference tolerance than the one used in the RDS algorithm to qualify confidence, meaning more or less 5 bpm or 5% from the true heart rate.

Different types of noises are modelled and can be applied to the signal. Example samples are presented on Fig. 6 (Appendix). They are all defined by a proportion interval, that corresponds to the interval in which to take the rate of original signal that the noisy signal will represent and a weight, which correponds to the sampling rate when randomly choosing the noises to appear in a signal, sampled from all available noises. List of the different signals is presented below:

- *EMG Noise:* Proportion Interval: [0, 5], weight: 1.5, `neurokit2.emg_simulate()` of duration of the epoch size to which to apply the noise to.
- *Powerline Noise:* Proportion Interval: [0, 90], weight: 1, sinusoid of amplitude 50 or 70Hz for respectively Europe or US powers. Set to 50Hz in our experiment.
- *Lead-off Noise:* Weight: 1.25, signal by itself. Corresponds to ECG electrode disconnected while device still on; white noise centered on 0 of standard deviation 1.
- *Low-R Noise:* No proportion needed as it transforms the full signal, weight: 3.25. Corresponds to abnormally low R-peak, leading to wrong heart rate detection; median filter of random kernel size in interval [0.06, 0.11], in range of the R-peak duration (0.08 to 0,1 depending on the heart rate).
- *Step Noise:* Proportion Interval: [0, 70], weight: 3corresponds to sudden movement; sigmoid function: $sigmoid = 1/1 \times \exp -\alpha * (x - \beta)$; with α the slope of the step and β the position of the step in the epoch, chosen randomly.
- *Motion Noise:* Proportion Interval: [0, 170], weight: 2.25, corresponds to movement, either due to activity of the subject or respiration; usage `neurokit2.resp_simulate()` using simple sinusoidal signal and respiration rate randomly chosen in interval [7, 21] (normal respiration rate between 12 and 16).

c) MIT Database: MIT dataset with systematic labelling: This database is similar to the Neurokit database except that we use the MIT-BIH Arrhythmia Database [5] instead of artificial clean ECG from the `neurokit` library. This database consists in clinical two-channel ambulatory ECG collected on both in and outpatients. Data is not labelled as clean and some samples even contains arrhythmias. Considering that the original dataset is big, even if it is noisy on some parts, it is mostly clean enough to be used as the clean template, similarly to `neurokit`-generated data. Hence, we neglect the noisiness of the original signal as long as we can derive an heart rate on it. Then, noises are added similarly to the Neurokit dataset and labels are obtained similarly.

d) Presentation of the experimental data: Training datasets vary depending on the model they are used for. Threshold-based univariate methods don't need any training set, density estimation methods require training set containing only samples of normality (clean) and supervised learning methods need training set with both clean and noisy data.

When possible, data were kept between the different models for a same dataset. Testing dataset can be kept for all models while training sets are similar between models of the same category. Tables II, III and IV present the composition of the testing datasets created for respectively the RDS dataset, Neurokit dataset and MIT dataset. Features are then calculated for each epoch of the dataset. Pairplot distribution of those features (log-transformed) are presented in Fig. 7 (Appendix) for the 3 datasets.

Signal		#Epochs (10sec)
Clean	997	360
	707	363
	779	363
	703	363
	656	363
	657	356
	668	363
	666	363
Lead_off	761	4898
Total	**9**	**7792**

Table II: Summary table for RDS dataset used in the experiments*. Presented data corresponds to the set from which the data can be fetched. It will constitute both the training and testing set. Signal column consists in the type of signal and the study id from staging server.*

Signal type	Clean	Noisy
Original	250	0
Lead_off	0	45
Motion	9	101
EMG	41	365
Powerline	159	351
Low_R	19	81
Step	22	201
Total	**500**	**500**

Table III: Summary table of proportion of each noise in the Neurokit testing dataset*. The same testing datasets are used for all models while the training datasets differ from one category of models to the other (density estimation, supervised learning or univariate feature-based). Original data are directly generated using the* `neurokit` *library. Combination of noises was added to the original signals to generate the other signal types. The number of epochs associated with each noises consequently doesn't add up to the total number of epochs as it corresponds to the number of epochs in which the noise is present, meaning one epoch containing multiple noises is counted multiple times.*

Signal type	Clean	Noisy
Original	250	0
Lead_off	0	38
Motion	19	73
EMG	46	319
Powerline	109	379
Low_R	42	61
Step	34	174
Total	**500**	**500**

Table IV: Summary table of proportion of each noise in the MIT testing dataset*. The same testing datasets are used for all models while the training datasets differ from one category of models to the other (density estimation, supervised learning or univariate feature-based). Original data is coming directly from the MIT database. Combination of noises was added to the original signals to generate the other signal types. The number of epochs associated with each noises consequently doesn't add up to the total number of epochs as it corresponds to the number of epochs in which the noise is present, meaning one epoch containing multiple noises is counted multiple times.*

3) Models: Different ways of computing the SQIs were compared. Each model outputs a *score* that corresponds to its SQI. Then, an adapted threshold-based classification is performed based on this score and evaluation metrics are computed to measure and compare the efficiency of the classification.

We distinguish 3 sorts of models. Threshold-based univariate models consist in taking one of the features as the score and find the best threshold to separate data, knowing the labels. Density estimation models, consist in approximating a "normality" distribution of what we consider usable data to classify both usable and unusable data depending on how well it fits to this approximation. Finally, supervised learning is found efficient to prevent too much assumptions on the data distribution, even if it is computationally more heavier.

Regarding the choice of the threshold for classification based on the SQI, we choose to take the threshold at the operating point of the ROC (Receiver Operating Curve). It is a good trade-off between a high specificity and a high sensitivity as it corresponds to the point of the ROC which is the closed to the (0,1)-corner [7]. To calculate it, we use the labels on the samples.

a) Feature-based classification: Even if this type of model has the main advantage to be simple, it is not flexible. We expect it to work well for a given type of noises but to be poorly efficient to classify all types of noises at once. Data consists in a set of both clean and noisy signals.

b) Density Estimation models: Density estimation walks the line between unsupervised learning, feature engineering, and data modeling [8]. Here, we compare two types of density estimation models. For both models, once the model is fitted to clean data, the distance between the samples and the distribution is evaluated using method `scores_samples`. This gives the weighted log probability for each sample. Then, classification is performed using the threshold obtained at the operating point of the ROC (Receiver Operating Curve), meaning we also need the labels on testing set. In order to learn a more robust distribution, we normalize the features that are used for fitting and validation.

Gaussian Mixture model (GMM): A Gaussian Mixture Model (GMM) is a probabilistic model that assumes all the data points are generated from the mixture of a finite number of Gaussian distribution with unknown parameters. Upon training, it learns the parameters that makes the Gaussian distributions fit the training data, which are only positively-labelled data. Consequently, in the case of noise detection, the rationale is to train the model on clean data only and subsequently predict the type of signal (noisy or clean) based on it fitting the distribution. We used `sklearn.mixture.GaussianMixture` with `n_components=1`, n_components being the number of Gaussian distribution used to fit the data and `n_init=10`, n_init being the number of initialisation performed (only the best is taken).

c) Kernel Density Estimation model (KDE): A Kernel Density Estimation model (KDE) estimates the probability density function of a set of data from the training set. Similarly to the GMM, we use only clean data to train the model to recognize a "normality" but it is

not assumed that the distribution is gaussian. We used `sklearn.neighbors.KernelDensity` with a gaussian kernel.

d) Supervised learning models: This type of model is based on learning from the features and the labels. For both models, once the model is fitted to clean data, the probabilities to be in each of the classes (clean or noisy) are evaluated using method `predict_proba`. The probability to be a clean signal is kept as the score for each sample. Then, the classification is performed using directly the method `predict` which chooses an optimal threshold and returns the corresponding estimated class.

e) Random Forest Classifier: The principle behind the Random Forest classifier is to decompose classification into multiple weak classifications, or trees, which all classify a different aspect of the complexity of the data. Each tree outputs a decision and the final decision is chosen using a majority vote. We used `sklearn.ensemble.RandomForestClassifier` with the defaults settings.

f) Logistic Regression: Logistic regression consists in using a logistic function to model a binary dependent variable used for classification. We used `sklearn.linear_model.LogisticRegression` with default parameters.

4) Evaluation metrics: When running an experiment we obtain a set of different evaluation metrics.

a) AUROC: We used the AUROC (area under the receiver operating characteristic) metric as a first way to evaluate the performance capacities of our models. First, we applied cross-validation, using `sklearn.model_selection.StratifiedKFold` with `n_fold=10` on the data so that for each fold, 1/10th of the data is not used. Then, the ROC (Receiver Operating Characteristic) curve is plotted using `sklearn.metrics.roc_curve` and the AUC (Area Under the Curve) computed with `sklearn.metrics.auc`.

b) Confusion Matrix and False Positive versus False Negative: Comparing the true labels to our predictions, we compute true positives, true negatives, false positives and false negatives with the threshold found at the operating point of the ROC. It provides information on precision and recall of the model.

c) Log-likelihood: The likelihood describes the joint probability of the observed data as a function of the parameters of the chosen statistical model. Hence for each observed sample, the likelihood assigns a probabilistic prediction to each class (here clean or noisy). The log-likelihood takes the logarithm of this probability. We used `sklearn.metrics.log_loss` method.

d) Precision and Recall: Precision corresponds to the part of signals that are classified as usable from the stack of truly usable signals, i.e. that are not wasted. In other terms, $Precision = \frac{TP}{TP+FP}$.

Recall corresponds to the part of signals that are truly usable from the stack of signals that are classified as usable, i.e. derived. In other terms, $Recall = \frac{TP}{TP+FP}$.

e) F-scores: The F-scores is a measure of accuracy of the model. The F_1-score, $F_1 = 2 * \frac{precision*recall}{precision+recall}$, computes the balanced harmonic mean between precision and recall. As we want to favor precision, we also compute a F_β-score with $\beta = 0.25$ and $\beta = 2$ using the following: $F_\beta = (1+\beta^2) * \frac{precision*recall}{\beta^2 precision+recall}$. We used method `sklearn.metrics.fbeta_score`. This metric is similar to the F_1 score but favors precision when β is smaller than one and recall when it is bigger.

f) Personalized loss: We started designing a loss that is specifically adapted to the MultiSense® solution. As it is still experimental, we calculate this loss manually on a few models. We perform a simplified risk analysis in the form of a Bayesian network, represented on Fig. 2. It includes multiple events that can happen throughout the patient's recovery journey, having an impact on this journey.

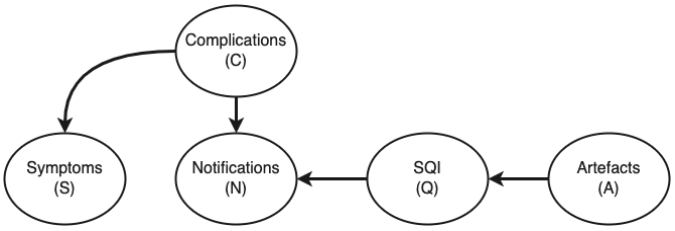

Figure 2: Bayesian network modeling events that can happen in the post surgery process and their inter-dependencies. *Events are represented as the nodes and influence of one event on another through a one-way arrow. Events are either directly linked to the health of the patient or to the performances of our MultiSense® solution.*

Events *C* and *S* are linked to the health of the patient, meaning they are fully independent from the MultiSense® performances. Previous literature on different colorectal surgery agree on a range of 12.7% to 30% of the patients that are readmitted [9][10]. We hypothesize that, for a patient who will have to be readmitted, only 50% of the recorded epochs will show complication (note that it is already a very conservative hypothesis, as soon as complications occur in the signal, the patient is brought back to the hospital for check up). Hence, we set $P(C) = 0.3 \times 0.5 = 0.15$. Then event *S*, *experience of symptoms by the patient* is dependent of event *C*. We suppose $P(S = 1|C = 1) = 0.9$, meaning most of the time complication will be perceptible, and $P(S = 1|C = 0) = 0.05$, meaning we consider a small part of the patients that will feel nocebo symptoms. We obtain the Conditional Probability Table (CPT) shown on Table V.

C	P(S=1$\|\|C$)	P(S=0$\|\|C$)
1	0.9	0.05
0	0.1	0.95

Table V: Conditional Probability Table (CPT) for event S*, the patients experiences symptoms, relative to event C, the patient presents complications.*

A second part of the network concerns outputs of our solution. Event *A* is *the signal presents artefacts*. Here we define artefacts as noise that makes the signal not safely derivable. Artefacts can be patient motions or powerline noise for instance, and we used the analysis performed on one of our clinical studies, *READASUR* (unpublished data), to set $P(A) = 0.15$. From that, the SQI algorithms, which we are

actually trying to improve with the following investigation, classifies the epoch as noisy or clean. Event Q is *the SQI algorithm classifies the epoch as clean*. Hence the associated CPT actually corresponds to the confusion matrix of our classification problem (correspondence shown on Fig. VI).

A	P(Q=1$\|\|A$)	P(Q=0$\|\|A$)
1	False Negative	True Negative
0	True Positive	False Positive

Table VI: Conditional Probability Table (CPT) for event Q, which corresponds to the confusion matrix of the models.

Event N is *the MultiSense® solution generates an out-of-bound notification*, meaning the derived physiological values are found to be out of normal ranges and an alarm is raised. It depends on C but also on Q and we reasonably hypothesis $P(N=1|Q=1,C=1)=0.9$, $P(N=1|Q=1,C=0)=0.05$ and $P(N=1|Q=0,C=0)=P(N=1|Q=0,C=1)=0$, obtaining the table presented in Table VII.

C	Q	P(N=1$\|\|C,Q$)	P(N=0$\|\|C,Q$)
1	1	0.9	0.1
0	1	0.05	0.95
0	0	0	1
1	0	0	1

***Table VII: Conditional Probability Table (CPT) for event N**, the MultiSense solution generates an out-of-bound notification, relative to event C, the patient presents complications and Q, the SQI algorithm classifies the epoch as clean.*

From those hypothesis, we obtain a fully completed Bayesian network at the exception of the CPT for event Q, as it corresponds to the performance of our models. To calculate the personalized loss for a given model, we complete the network with the confusion matrix values for the given model. Then, by sampling the network a high number of times, we obtain happening probability for each scenario. Here, a scenario refers to a sampled combination of events. Then, by manually evaluating the cost of each scenario regarding safety of the patient and performances of our device (see Table XIII), we can weight each scenario's probability. Finally, we sum the resulting terms to obtain a cost. The cost represents, in a very simplified way, the added value of our device and so we expect to get a negative cost.

We judge it more serious to provide wrong information than not to provide any as the first one could delay a deep investigation when a patient report feeling bad. Hence, we favor precision rather than recall when evaluating the performances.

B. SQIs for PPG

1) Riemannian Potato method: Artefacts are detected using the distance between covariance matrix of each epoch to the mean covariance matrix. For that, we use the `pyriemann` library [11]. The covariance matrix between the two PPG channels and the ECG over each epoch is computed using the `pyriemann.estimation.Covariances` class. Classification is then performed using the `pyriemann.clustering.Potato` class, an artefact detection method working similarly to k-means. Due to the geometrical properties of covariance matrices, the method uses a Riemannian metric to compare the covariance matrices to an average covariance matrix. It iteratively estimates the centroid of clean signal by rejecting every sample that is too far from it at each trial. Then, it computes a z-score for each sample, designating the probability to be in the normal distribution. We set threshold to discriminate z-score at 2.65 for all experiments.

2) Dataset: The method was tested in 3 steps. In order to validate its performances, we first compared discrimination capabilities to performances of the existing SQI for PPG. Hence, we used the model on clean and burn-in data from *staging* and *trial*. Labeling of these data was performed mostly on ECG signals, hence some mildly noises, mostly motion noises, in the PPG signals are present.

Data used from the servers are presented in Table VIII.

Signal		Start Offset	Stop Offset
Clean	656	136799	140429
	997-1	38163	41763
	997-2	45158	48758
	707-1	115994	119624
	707-2	130030	133660
	703	34337	37967
	779	47525	51155
	657	130977	134607
	666	33516	37146
Burn-in	761	0	7500
	762	0	7500
	763	0	7500
	767	0	7500
	768	0	7500

***Table VIII: Summary table for RDS dataset used in the experiments**. Clean data was used for training. Burn-in data was used to validated performances of the mode on burn-in compared to existing SQIs.*

In order to assess performances on more complex noises such as shifts or venous modulation, we use hand-labelled selected snippets of signals from RDS. Data consists in the two PPG channels and the ECG signal.

All data is filtered. The preprocessing pipeline is described below:

- PPG baseline removal: use `algorithms.display.filter_raw_ppg`.
- PPG Median filtering: `scipy.ndimage.median_filter` with `size=15` and `mode='wrap'`, similarly to pre-processing in the Algorithm component to remove potential 'spikes' in the raw PPG signal caused by power supply noise.
- PPG interpolation: Sampling frequency from 64Hz to 256Hz, which is sampling frequency of the ECG signal.
- ECG baseline removal: use `algorithms.display.filter_raw_ecg`.
- ECG Low-pass filtering: same low-pass filter as in `algorithms` with cut-off frequency of 3Hz to smooth the ECG peaks and make it more similar to PPG channels.
- ECG and PPG normalization: basic normalization per channel.

3) Experiments:

a) Leave-One-Study-Out (LOSO) on clean data: LOSO consists in training on clean studies except one and testing on this last study. Process is presented on Fig. 6. As testing is performed on a brand-new study, we can assess that the model is not overfitting on the training data.

b) Validation on Burn-in data: Training is done on clean dataset presented in the first part of Table VIII and validation

Figure 3: Leave-One-Study-Out (LOSO) principle. *Blue: studies used for training. Orange: study used for testing. Each line corresponds to a new model, trained with the blue studies and validated on orange study. Performances for each testing set are then concatenated (right column) to be evaluated.*

classification is performed on the burn-in sequences in the second part of Table VIII.

c) Validation on specific data sequences: Training is done on the clean dataset presented in the first part of Table VIII. Then, the model is used to classify multiple snippets of signal, specifically labelled to contain interesting features such as shifts or venous modulation. Those snippets and the type of noise they contain are listed in Table XII (Appendix).

III. Results

A. SQIs for ECG

Different metrics were computed for all models and datasets. Complete results are presented in Table XI. The ROC curves as well as the confusion matrix were computed for all models. We show ROC curve (Fig. 4.a) and confusion matrix (Fig. 4.b) only for Kernel Density Estimation model on the Neurokit database.

B. SQIs for PPG

Results of the LOSO are presented in Table IX. They are compared to the results obtained on same data using the current SQI algorithm from RDS. Example of a result of the classification on clean data shown with the corresponding signal is shown on Fig. 5. Results of the validation on burn-in data are presented in Table X. Finally, results for each sequence of specific data are presented in Table XII. As the current algorithm does not discriminate shifts, no comparison is presented. The data are sorted by label and performances of the algorithm.

IV. Discussion

A. SQIs for ECG

Finding a general and efficient SQI is challenging as the models performances depend on a large set of parameters that could make our final conclusions vary greatly. The main bias is our confidence in the labels. For that we designed three different datasets on which we ran all the models and we are looking for an ambivalent model that performs well for all the databases. Another bias comes with evaluating the SQI models results. We want a metric that evaluates the models in the frame of the specific problematic of the MultiSense® solution. For that we use divers techniques and we make sure that the SQI does well for all metrics. Moreover, we design our own evaluation metric based on the clinical risks that the MultiSense® device encounters. That allows us to favor the best SQI in the precise context of our device.

Study	Spo2_CL (%)	Riemann_SQI (%)
656	96,81 %	96.80%
703	97,14 %	96.13%
707-1	99,20 %	3.31%
707-2	99,94 %	99.11%
779	96,53 %	87.85%
997-1	99,17 %	97.55%
997-2	98,42 %	97.10%
666		95.25%
Global	**100% correct (8/8)**	**87.5% correct (8/9)**

Table IX: Performances comparisons between current SQI implementation and Riemannian potato on testing data only, using LOSO to train the models. *Riemannian_SQI column corresponds to the rate of epoch that the model classified as clean in the full sample. We consider that the algorithm is correct (in green) when it classifies not more than 10% of the epochs as clean in noisy samples and at least 80% of the epochs as clean in clean samples. Global values correspond to the rate of sequences that were qualified correctly, following those rates.*

Study	Spo2_CL (%)	Riemann_SQI (%)
761	0,03 %	0.0%
762	0,05 %	0.0%
763	2,13 %	0.001%
767	3,41 %	0.0%
768	0,18 %	0.0%
Global	**100% corect (5/5)**	**100% correct (5/5)**

Table X: Performances comparisons between current SQI implementation and Riemannian potato on burn-in data. *Riemannian_SQI column corresponds to the rate of epoch that the model classified as clean in the full sample. We consider that the algorithm is correct (in green) when it classifies not more than 10% of the epochs as clean in noisy samples and at least 80% of the epochs as clean in clean samples. Global values correspond to the rate of sequences that were qualified correctly, following those rates.*

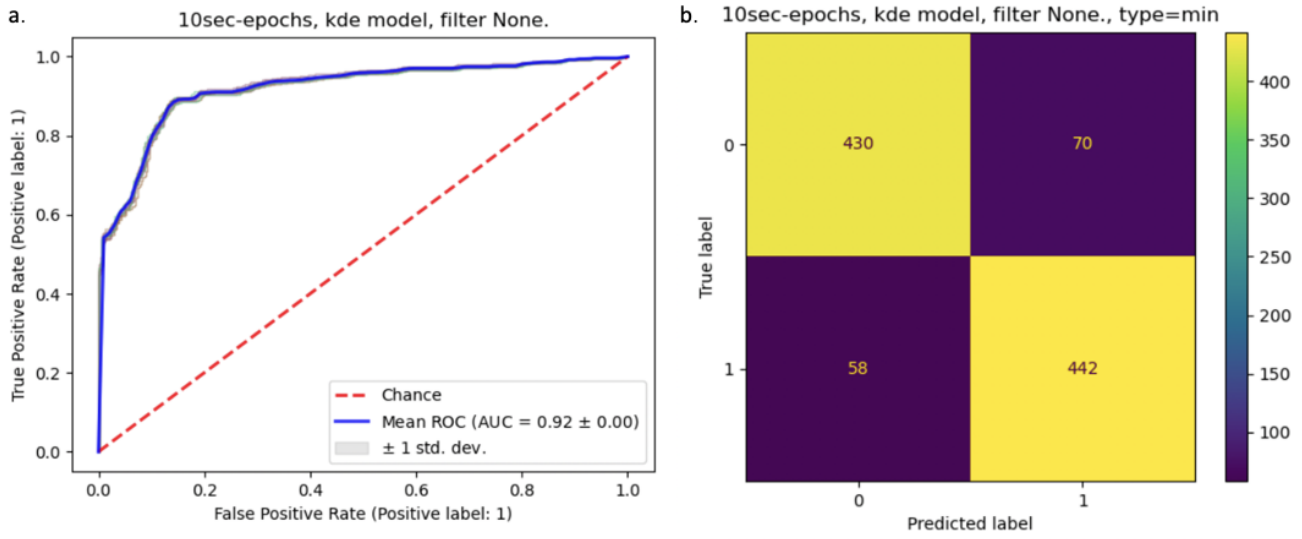

Figure 4: Results for Kernel Density Estimation model on Neurokit dataset. a. *ROC (receiver operating characteristic curve, False Positive rate versus True Positive rate) and corresponding Area Under the Curve (AUC). Blue line is the mean ROC obtained after bootstrapping and pale curves correspond to each fold of the bootstrapping, dashed red line represents a random classification, as a comparison, grey area represents a confidence band of ± 1 around mean ROC.* ***b.*** *Confusion matrix, showing True Positive, False Negative, False Positive and True Negative.*

B. Results analysis

Analysis of the results in Table XI lead to a number of observations on the compared models.

We observe that all models perform really good on the RDS dataset ($AUROC \geq 0.97$ for all). However, it is a really simple dataset that only contains two types of signal: clean signals and lead-off signals. Hence, for now if it cannot help to discriminate between the different models, it assess that the models work at discriminating between clean and lead-off data. It would be good to add other hand-labelled snippets, corresponding to different noises to test the models on fully real data. Models are expected to perform better when trained on Neurokit dataset than MIT dataset as the data that we classify as clean in MIT dataset might actually be noisy. And indeed, we see that, looking at the AUROC, most of the models (at the exception of powerline ratio, which is actually performing badly on both) perform better on Neurokit dataset.

Regarding powerline-ratio-based classification being good on RDS dataset ($AUROC = 0.99$) but not on the other datasets ($AUROC = 0.5$ and $AUROC = 0.53$ for respectively Neurokit and MIT), this feature was especially designed to discriminate lead-off signals, which tend to contain a lot of powerline frequencies. Hence, the model is efficient on lead-off noise but not so much on other noises, explaining bad performances when datasets contain different noises. And actually it is confirmed when looking at Fig. 7.b and d. in which distribution of the signals containing powerline noises (in green) seems to be better separated from the rest of the dataset when looking at powerline ratio line (one before last from the top and from the left). More generally, as expected, the univarate feature-based models (in blue) are not as good as the machine learning models.

Comparing density estimation models, Kernel Density estimation gets better performances as Gaussian Mixture model. It makes sense as GMM makes the strong assumption that the normality distribution is Gaussian, which is in practice not the case. KDE isn't making assumptions on the distribution of the data, which allows it to get closer to the true distribution and by such showing better classification performances. Both of them having satisfactory performances on MIT data is good. It means that we can use these models on data that is not strictly clean, i.e. as part of an online experiment in the RDS pipeline to treat incoming signals. It is important to note that the classification method uses labels to define the threshold and separate scores. Consequently it is not implementable online as it is and the obtained metrics are optimal metrics with a threshold adapted to the dataset. However, as we normalize the features to train the models, we expect the optimal threshold for each new dataset to always be found around the same value. After generating multiple datasets, we observed that indeed the optimal threshold is always around -17.5 for RDS dataset, -17.9 for Neurokit dataset and -17.5 for MIT dataset. Further work needs to be done to calibrate a global threshold more precisely but we hypothesize that similar performances would be reached by using a fixed threshold set at -17.6 for all dataset.

Supervised models are performing better than density estimation models and Random Forest classifier manages to be the most consistent across datasets. However, as said before KDE also has good performances. We prefer to select KDE. It has good performances and can be used online, without labelled data. Consequently, it is more convenient to

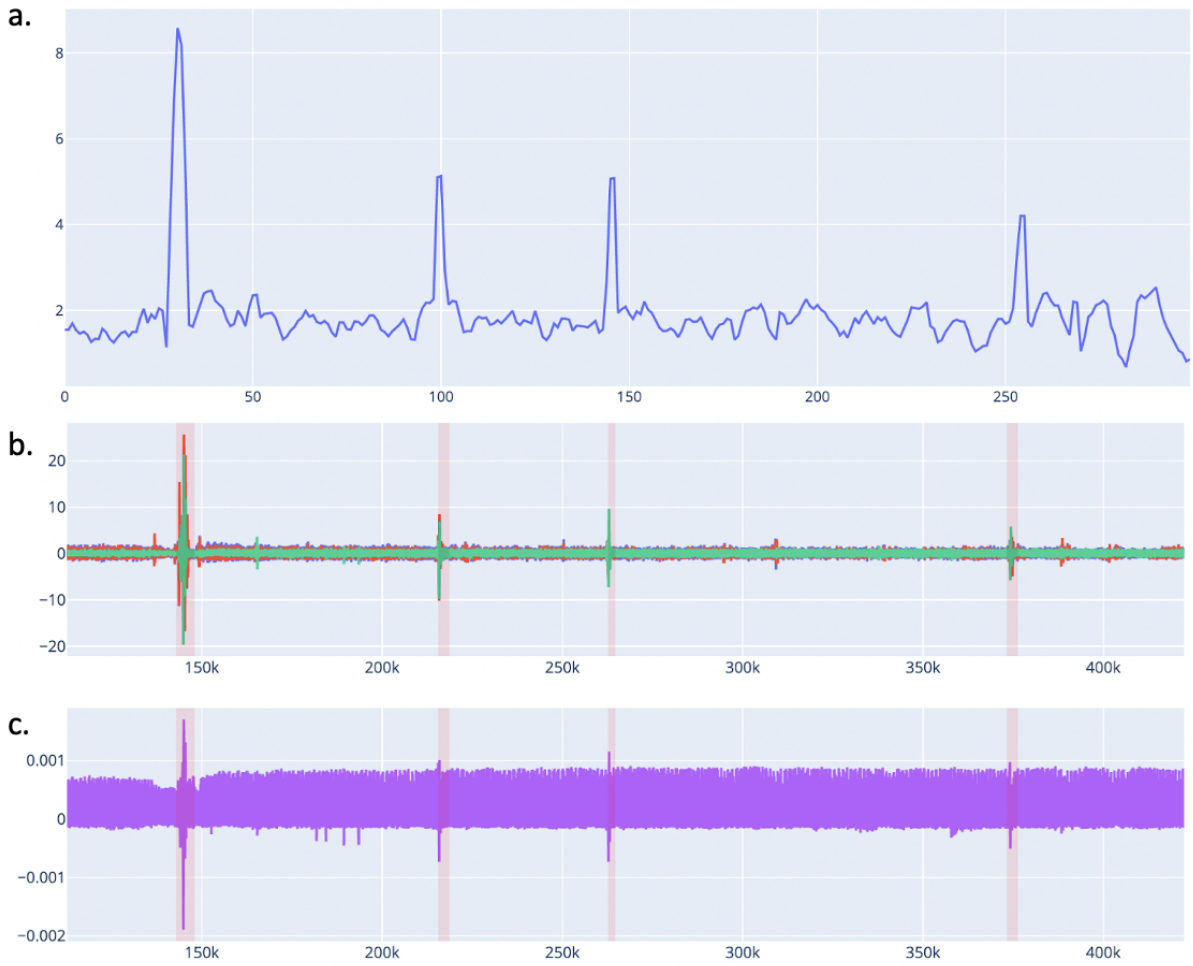

Figure 5: Classification results using Riemannian Potato model on a clean segment signal of study 656 (RDS staging - 300 epochs). a. *Distance between the individual covariance matrices (for each epoch) and the mean covariance matrix.* ***b.*** *Results of the classification, with red bands meaning the algorithm classified the segment as noisy. Signals correspond to, in blue, infra-red PPG signal, after baseline removal, median filtering and normalization, in red, red PPG signal, after baseline removal, median filtering and normalization, in green ECG signal, after baseline removal, low-pass filtering and noramlization and* ***c.*** *in purple, ECG signal, after baseline removal.*

implement in the current set-up of the RDS pipeline.

C. Pros and cons of our different datasets

Each database have its pros and cons.

- RDS Database has the clear advantage to be real clinical data, obtained directly with the device we aim at improving. Consequently, the labelled noises are close to what the algorithm will encounter in an everyday usage, compared to the more ideal artificial noises that we apply on the tow other datasets. However, it is hand-labelled, meaning a lower confidence in the labels, fewer types of noises (only clean and lead off labelled in a sufficient amount to be used in our models) and fewer data to work with.
- On the contrary, using the Neurokit dataset, we can generate as much data as needed with as many noises as we can design artificially. The set is much more flexible and controllable. The labelling technique is also more sure: it is an automatized process, without the uncertainty of a hand-labelled dataset. Moreover, it is based on the quality of the heart rate derived on the signal, using the same criteria as in our algorithms. It means that we directly label the signal on the accuracy of the information derived from the ECG signal.
- Finally, the MIT Dataset as both the last datasets' advantages. It is based on real clinical data, the original MIT-BIH dataset is big enough to have a large and varied dataset and noises are also added artificially. Regarding the initial quality of the signals from the MIT-BIH dataset, we do the hypothesis that the original data is clean because we consider that the original dataset is reasonably big enough so that random picking will mostly provide us with clean signals. However, data could be noise and could even contain arrhythmia.

D. Evaluation metrics for ECG signals and risk analysis

To evaluate and compare the different models, we computed different evaluation methods, designed to be adapted to the requirements of our algorithm so that it adds value to

the device and taking into account the trade-off between availability and performance. The whole complexity is to accurately define what is relevant in the context of our very specific problematic. Regarding the clinical problematics linked to the usage of the device, we want to be sure that the data shown to the practitioner and derived to get a heart rate for ECG signals is clean. It means we cannot tolerate false positive (noisy epochs classified as clean): we want a high precision. On the other hand, having a high sensitivity also means having a significant number of false negative (clean data discarded as being noisy). That means a part of the good and usable data could be falsely discarded and not shown to the practitioner. If it results in no data for a long period of time, it might end up slowing down the diagnosis of a condition. Hence, high recall is also required.

Results of the personalized loss for a few models seem to make sense as it gives the same conclusions as the other metrics when comparing the models. However, we realized that a covariance matrix with no true negative and no false negative (i.e. personalized loss for powerline ratio model on Neurokit, (1,499,0,500)) gives a really good loss while it is only classifying all epochs as positive. The obvious, but disturbing, conclusion here is that as True Positive is most beneficial category and False Positive happen really rarely, it is actually more beneficial to classify noises as clean, to be sure to classify clean data as clean to use it. It would make this whole project useless. However, practically, it seems dangerous derive information from noisy data. Hence, for now, we won't consider the limit cases and will only use this loss in addition to the more traditional evaluation metrics, as a mean of comparison between the models that are evaluated.

E. SQIs for PPG

Tables IX and X show that the Riemannian potato method performs as good as current algorithms on clean and burn-in data. Performances on burn-in signals are better than the current SQI. For clean signals, we observe that classification rate as clean is always lower than current SQI. However, looking at classification along with the corresponding signal (Fig. 5), this lower rate might be explained by a higher sensitivity to motion noises.

Having a high sensibility to motion noises might be problematic. Currently, motion noises are removed when activity reaches a certain threshold. By being more sensitive to noises, availability might be lower and we might discard data that could still be properly derived. We need to select a classification threshold high enough not to be sensitive to mildly noisy epochs.

Regarding the selected sequences, we observe a globally good detection of clean signal (5 out of 11). The misclassified sequences can, in part, be explained by the high sensitivity to noise. Indeed, 3 sequences out of the 6 misclassified are considered as failed while the clean rate is still higher than 50. Moreover, low pass filtering on the ECG signal provokes a border effect which adds some noise to the original signal. Globally, the model also detects shifts accurately when shifts happen between PPG channels (9 out of 10). Finally, there is no detection of venous modulation (1 out of 13). Indeed venous modulation corresponds to a complete phase shift, making the correlation between the signals similar to a normal signal.

Further work should be performed in order to discriminate venous modulation better. For that, cospectral matrix could be used as descriptors of the epochs, instead of covariance matrix. They consist in covariance matrices by frequency range. PPG signals being composed of both cardiac and respiratory components, happening at different frequencies, we could have two cospectral matrices, one between 0.1 and 0.5Hz for the cardiac component and between 0.5 and 3Hz for the respiratory component. We hypothesize that it would make the model more robust as the features are more specific. Another area of investigation is manual shifting of one of the PPG signal in order to identify venous modulation. By dephasing one of the PPG signals and learning from this new dataset, we expect that we could detect venous modulation.

REFERENCES

[1] Zhidong Zhao and Yefei Zhang. "SQI Quality Evaluation Mechanism of Single-Lead ECG Signal Based on Simple Heuristic Fusion and Fuzzy Comprehensive Evaluation". In: *Front. Physiol.* 0 (2018). ISSN: 1664-042X. DOI: 10.3389/fphys.2018.00727.

[2] Christina Orphanidou et al. "Signal-quality indices for the electrocardiogram and photoplethysmogram: derivation and applications to wireless monitoring". In: *IEEE J. Biomed. Health Inf.* 19.3 (May 2015), pp. 832–838. ISSN: 2168-2208. DOI: 10.1109/JBHI.2014.2338351. eprint: 25069129.

[3] Alexandre Barachant, Anton Andreev, and Marco Congedo. "The Riemannian Potato: an automatic and adaptive artifact detection method for online experiments using Riemannian geometry". In: *TOBI Workshop lV* (Jan. 2013). URL: https://www.researchgate.net/publication/280701015_The_Riemannian_Potato_an_automatic_and_adaptive_artifact_detection_method_for_online_experiments_using_Riemannian_geometry.

[4] Paolo Castiglioni et al. "Cepstral based approach for online quantification of ECG quality in freely moving subjects". In: *ResearchGate* 38 (Jan. 2011). URL: https://www.researchgate.net/publication/241627314_Cepstral_based_approach_for_online_quantification_of_ECG_quality_in_freely_moving_subjects.

[5] G. B. Moody and R. G. Mark. "The impact of the MIT-BIH Arrhythmia Database". In: *IEEE Eng. Med. Biol. Mag.* 20.3 (May 2001), pp. 45–50. ISSN: 1937-4186. DOI: 10.1109/51.932724.

[6] Dominique Makowski et al. “NeuroKit2: A Python toolbox for neurophysiological signal processing”. In: *Behavior Research Methods* 53.4 (Feb. 2021), pp. 1689–1696. DOI: 10.3758/s13428-020-01516-y. URL: https://doi.org/10.3758%2Fs13428-020-01516-y.

[7] Bowen Song et al. “ROC operating point selection for classification of imbalanced data with application to computer-aided polyp detection in CT colonography”. In: *International journal of computer assisted radiology and surgery* 9.1 (Jan. 2014), p. 79. DOI: 10.1007/s11548-013-0913-8.

[8] *2.8. Density Estimation*. [Online; accessed 24. Nov. 2021]. Nov. 2021. URL: https://scikit-learn.org/stable/modules/density.html.

[9] Johanna Van Butsele et al. “Readmission after rectal resection in the ERAS-era: is a loop ileostomy the Achilles heel?” In: *BMC Surg.* 21.1 (Dec. 2021), pp. 1–9. ISSN: 1471-2482. DOI: 10.1186/s12893-021-01242-y.

[10] N. K. Francis et al. “Factors predicting 30-day readmission after laparoscopic colorectal cancer surgery within an enhanced recovery programme”. In: *Colorectal Dis.* 17.7 (July 2015), O148–O154. ISSN: 1462-8910. DOI: 10.1111/codi.13002.

[11] *pyRiemann: Biosignals classification with Riemannian Geometry — pyRiemann 0.2.8.dev documentation*. [Online; accessed 4. Jan. 2022]. Dec. 2021. URL: https://pyriemann.readthedocs.io/en/latest.

APPENDIX

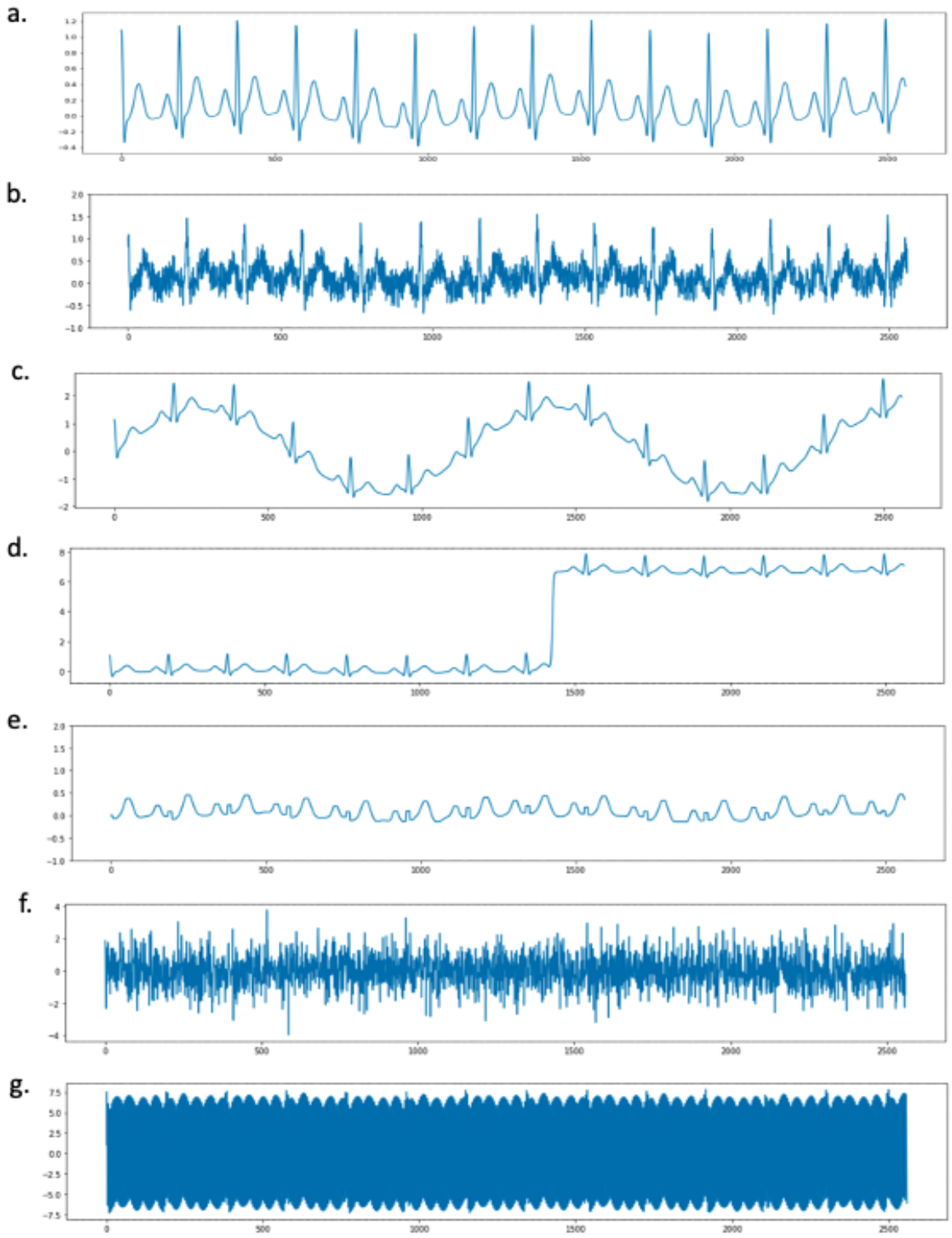

Figure 6: ***a.*** *Clean 10-seconds epoch of 80bpm,* ***b.*** *Corresponding noisy signal with EMG noise of 150% of the energy of the original signal. Respective heart rates computed using the RDS heart rate detection algorithm are 80 and 86bpm. The difference in resulting calculation is sufficient to indicate that the noisy snippet can be labelled as noise,* ***c.*** *Corresponding noisy signal with motion noise of 500% of the energy of the original signal,* ***d.*** *Corresponding noisy signal with step noise of 5000% of the energy of the original signal.* ***e.*** *Corresponding noisy signal with low R noise,* ***f.*** *Lead off noise,* ***g.*** *Corresponding noisy signal with 50Hz-powerline noise of 5000% of the energy of the original signal.*

Model	RDS	Neurokit	MIT Physionet
GMM	Precision: 0.77 Recall: 1.00 AUROC: 0.99 F1-score: 0.87 F0.25-score: 0.78 Log-Likelihood: 1.74	Precision: 0.82 Recall: 0.89 AUROC: 0.90 F1-score: 0.86 F0.25-score: 0.83 Log-Likelihood: 5.18	Precision: 0.59 Recall: 0.94 AUROC: 0.54 F1-score: 0.73 F0.25-score: 0.61 Log-Likelihood: 12.23
KDE	Precision: 1.00 Recall: 1.00 AUROC: 0.99 F1-score: 1.00 F0.25-score: 1.00 Log-Likelihood: 0.02	Precision: 0.86 Recall: 0.88 AUROC: 0.92 F1-score: 0.87 F0.25-score: 0.86 Log-Likelihood: 4.42 Personalized loss: -3.58	Precision: 0.93 Recall: 0.63 AUROC: 0.85 F1-score: 0.75 F0.25-score: 0.90 Log-Likelihood: 7.29 Personalized loss: -2.46
Random Forest	Training F0.25-score: 1.00 Training Log-Likelihood: 0.01 Precision: 1.00 Recall: 1.00 AUROC: 0.99 F1-score: 1.00 F0.25-score: 1.00 Log-Likelihood: 0.03	Training F0.25-score: 0.98 Training Log-Likelihood: 1.55 Precision: 0.92 Recall: 0.86 AUROC: 0.95 F1-score: 0.89 F0.25-score: 0.92 Log-Likelihood: 3.70 Personalized loss: -3.49	Training F0.25-score: 0.93 Training Log-Likelihood: 2.67 Precision: 0.93 Recall: 0.67 AUROC: 0.88 F1-score: 0.78 F0.25-score: 0.91 Log-Likelihood: 6.63 Personalized loss: -2.50
Log regression	Training F0.25-score: 1.00 Training Log-Likelihood: 0.01 Precision: 1.00 Recall: 1.00 AUROC: 0.99 F1-score: 1.00 F0.25-score: 1.00 Log-Likelihood: 0.03	Training F0.25-score: 0.90 Training Log-Likelihood: 3.70 Precision: 0.94 Recall: 0.88 AUROC: 0.95 F1-score: 0.91 F0.25-score: 0.93 Log-Likelihood: 3.04 Personalized loss: -3.65	Training F0.25-score: 0.85 Training Log-Likelihood: 6.42 Precision: 0.81 Recall: 0.80 AUROC: 0.87 F1-score: 0.80 F0.25-score: 0.81 Log-Likelihood: 6.74 Personalized loss: -3.19
ν-SVM with ν = 0.5	Training F0.25-score: 1.00 Training Log-Likelihood: 0.01 Precision: 1.00 Recall: 1.00 AUROC: 0.99 F1-score: 1.00 F0.25-score: 1.00 Log-Likelihood: 0.03	Training F0.25-score: 0.79 Training Log-Likelihood: 5.78 Precision: 0.82 Recall: 0.90 AUROC: 0.93 F1-score: 0.86 F0.25-score: 0.83 Log-Likelihood: 5.01	Training F0.25-score: 0.91 Training Log-Likelihood: 6.63 Precision: 0.90 Recall: 0.70 AUROC: 0.85 F1-score: 0.79 F0.25-score: 0.89 Log-Likelihood: 6.46
Std	Precision: 0.87 Recall: 0.99 AUROC: 0.98 F1-score: 0.93 F0.25-score: 0.88 Log-Likelihood: 1.80	Precision: 0.71 Recall: 0.94 AUROC: 0.86 F1-score: 0.81 F0.25-score: 0.73 Log-Likelihood: 7.50	Precision: 0.91 Recall: 0.63 AUROC: 0.84 F1-score: 0.74 F0.25-score: 0.89 Log-Likelihood: 7.46
Cepstral Distance	Precision: 1.00 Recall: 0.87 AUROC: 0.97 F1-score: 0.93 F0.25-score: 0.99 Log-Likelihood: 1.62	Precision: 0.83 Recall: 0.85 AUROC: 0.84 F1-score: 0.84 F0.25-score: 0.83 Log-Likelihood: 5.56	Precision: 0.58 Recall: 0.99 AUROC: 0.61 F1-score: 0.73 F0.25-score: 0.59 Log-Likelihood: 12.68
Powerline ratio	Precision: 1.00 Recall: 1.00 AUROC: 0.99 F1-score: 1.00 F0.25-score: 1.00 Log-Likelihood: 0.03	Precision: 0.50 Recall: 1.00 AUROC: 0.50 F1-score: 0.67 F0.25-score: 0.52 Log-Likelihood: 17.24 Personalized loss: -4.04	Precision: 0.50 Recall: 1.00 AUROC: 0.53 F1-score: 0.67 F0.25-score: 0.52 Log-Likelihood: 17.27

Table XI: Results of the ECG SQIs evaluation. *Performances for each model trained on each dataset. Models are sorted by categories, with density estimation models in yellow, supervised learning models in purple and univariate threshold-based models in blue.*

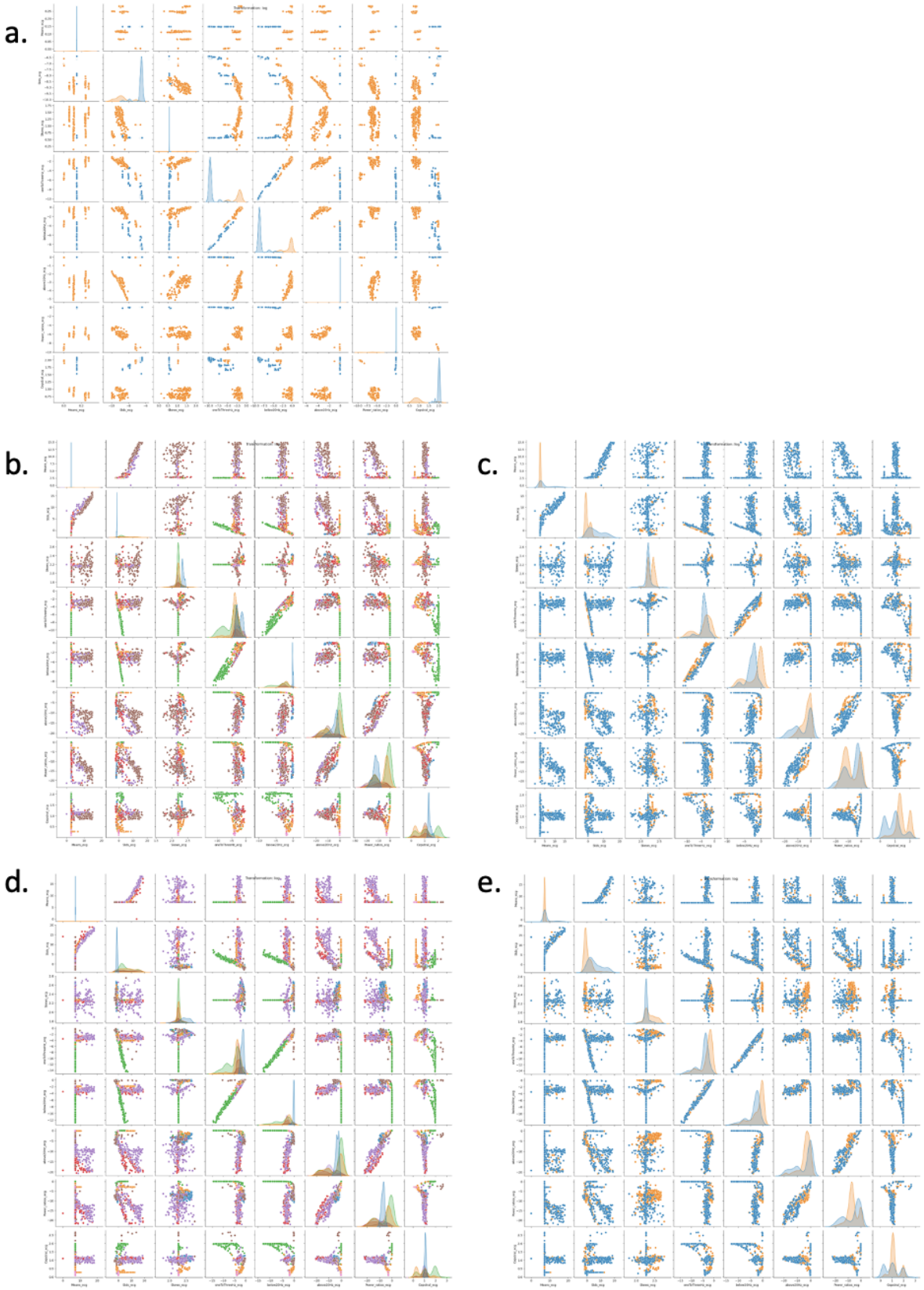

Figure 7: Pairplot distribution of the features by signal type and label for each dataset. For all pairplots, on the y-axis from top to bottom and x-axis from left to right, features are mean, standard deviation, skewness, power ratio between 1Hz and 3Hz, power ratio below 20Hz, power ratio above 20Hz, powerline ratio and cepstral distance to white noise. a. *Pairplot distribution by signal type (orange: lead-off signal (noisy), blue: clean signal) for RDS dataset. It is composed of only 2 signals so the pairplot per label is equivalent.* ***b.*** *Pairplot distribution by signal type (blue: original, orange: EMG noise, green: powerline noise, red: low-R noise, purple: motion noise, brown: step noise, pink: lead-off noise) and* ***c.*** *by label (blue: noisy, orange: clean) for Neurokit dataset.* ***d.*** *Pairplot distribution by signal type (blue: original, orange: EMG noise, green: powerline noise, red: motion noise, purple: step noise, brown: low-R noise, pink: lead-off noise) and* ***e.*** *by label (blue: noisy, orange: clean) for MIT dataset.*

Sample	Type of signal	Label	Riemann_SQI
581-5	Shift	0	0.71
581-4	Good → Bad	0	0.92
581-3	Venous Modulation	0	0.99
581-2	Venous Modulation	0	0.93
579-1	Venous Modulation	0	0.90
579-3	Venous Modulation	0	0.95
579-6	Venous Modulation	0	0.96
579-4	Venous Modulation	0	0.99
579-5	Venous Modulation	0	0.95
752-3	Venous Modulation	0	0.99
752-1	Venous Modulation	0	0.99
752-5	Venous Modulation	0	0.99
752-6	Venous Modulation	0	0.99
752-7	Venous Modulation	0	0.90
592-3	Deformed	0	0.99
TA-2	Deformed	0	0.95
426-2	Shift	0	0.0
ExampleSevereVenousAfterPositionChange	Venous Modulation	0	0.01
ExampleGoodToBad	Good → Bad	0	0.08
ExampleShiftedPhase	Shift	0	0.07
ExampleSevereShiftvenousMod	Shift	0	0.16
TC-2	Shift, deformed	0	0.00
ExampleRedShifted	Shift	0	0.07
TB-3	Shift, distortion	0	0.03
TB-2	Shift	0	0.05
492-2	Shift	0	0.03
TA-3	Deformed	0	0.0
TC-3	Shift, deformed	0	0.00
592-1	Shift	0	0.01
TC-1	Good	1	0.56
TA-1	Good	1	0.00
419-1	Good	1	0.51
426-1	Good	1	0.0
579-2	Good	1	0.75
ExampleGood	Good	1	0.65
752-2	Good	1	0.95
752-4	Good	1	0.95
TB-1	Good	1	0.88
592-2	Good	1	0.96
581-1	Good	1	0.99
Global		**27.75% clean (11/40)**	**47.5% correct (19/40)**

Table XII: Results of the Riemannian Potato model on sequences of interest. *Threshold set to 2.65, data containing both clean and noisy samples. Label is 0 for noisy data (in yellow) and 1 for clean data (in blue). Type of signal corresponds to the main interest in the sequence but that doesn't mean that the sequence does not contain motion noise for instance. Riemannian_SQI column corresponds to the rate of epoch that the model classified as clean in the full sample. We consider that the algorithm is correct (in green) when it classifies not more than 10% of the epochs as clean in noisy samples and at least 80% of the epochs as clean in clean samples.*

Complication	Symptomatic	Notifications	SQIs	Artefacts	Loss
0	1	1	1	0	0
0	0	1	1	0	0
0	1	0	1	0	-3
0	0	0	1	0	-4
0	1	0	0	0	4
0	0	0	0	0	0
0	1	1	1	1	4
0	0	1	1	1	3
0	1	0	1	1	4
0	0	0	1	1	0
0	1	0	0	1	0
0	0	0	0	1	0
1	1	0	0	1	0
1	0	0	0	1	0
1	1	1	1	1	0
1	0	1	1	1	0
1	1	0	1	1	12
1	0	0	1	1	14
1	1	1	1	0	-12
1	0	1	1	0	-14
1	1	0	1	0	0
1	0	0	1	0	0
1	1	0	0	0	8
1	0	0	0	0	10

Table XIII: Losses for each scenario. *Negative losses designate scenarii that add value to the product and positive losses are scenarii in which the device is problematic. Losses are set to assess benefices of the SQI algorithm only, not the notifications system for example. Hence, if the process needs to be reused for a different part of the MultiSense solution, one would have to adapt the losses.*