Report: PredMAIn – T1.4 PdM-Algorithms

Topic:	Prototype of PdM-Workflow on Cycle-Sensor Data (Corrected)
Date, Time	14.Nov 2022 10:00 - 11:00
Place:	Online
Attending:	SCCH (F.S.) – COMPAS (
Author:	F. Sobieczky, A. Karetnikov
Distributed to:	COMPAS, Intemac, SCCH

Contents

Summary: *A workflow for the specific case of process monitoring data which contains cycles repeating over a long time is presented. The method relies on predefined features ideally used as health-indicators which are aggregated over the cycles. Anomaly detection, HI-classification and RUL-prediction is then performed on the time series of aggregated values indexed by the cycles. For all three tasks, modern machine learning models are used to assist their completion.****Apart from the predictions of health state***

1 Anomaly Detection

To prepare the data for applying modern machine learning models they must have so called **labels**, indicating whether they belong to a healthy or unhealthy production state.

For this purpose, **anomaly detection** methods can be used. In the present context it refers to the detection of progressions of time series in a way which is **not typical** with regards to observations of the time series' values on some time interval in the past.

Definition: *'Predict and Compare'* – When on a defined interval of values in 'the present' (i.e., from the present point t backwards to a time $t-m$, after which a change in the data should be detected) the observations seem to have an underlying probability distribution significantly deviating from another time interval lying further in 'the past', then it is decided that an anomalous trend has been added to the systematic part of the signal and it is labelled 'unhealthy'. Otherwise, it is labelled 'healthy'. Note that the windows I and J are 'sliding' (traveling with t).

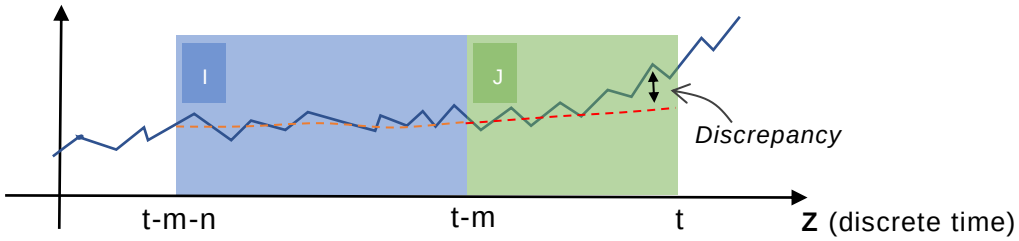

Figure 1: *Schematics of the windows I ('past') and J ('present', or 'recent past') acting as the domains of the input and output sequences of the ML model, respectively. The model predicts the trend (dashed red) on J from the observed trend (dashed orange) on I. A discrepancy between the two indicates a change in the trend from the predicted one.*

Notation:

Look-back window: $I = \{t-n-m+1, \dots, t-m\} \subset \boldsymbol{Z}$,

Prediction window: $J = \{t-m+1, \dots, t\} \subset \boldsymbol{Z}$,

Current point in time: $t \in \{1, \dots, n\}$,

ML-model: $F_t: \{v: I \to \boldsymbol{R}\} \to \{w: J \to \boldsymbol{R}\}$, where $\boldsymbol{Z}, \boldsymbol{R}$ are the integers and real numbers, respectively.

We assume, that at each point in discrete time $t \in \boldsymbol{Z}$, an ML model F_t can map finite sequences of values on I (the 'look-back window') to a sequence of predicted values on the finite sequence of values on J (the 'prediction window'). The function F_t can be any ML model (Linear Regression, ARIMA, Support Vector Machine, Random Forest, Deep Learning model, etc.), with the ability to forecast the progression of observed values reasonably into the future.

Powerful such models are recursive neural networks, such as LSTMs, as they detect trends in the systematic part of the input sequence and extrapolate them in prediction result. Linear models which also have this capability are ARIMA models.

The Predict & Compare method is an online change point detection method assisted by a machine learning (or 'AI') method.

Other *classical* online change point detectors in time series, such as the CUSUM control chart (Page-Hinkley test, [][][]), online trend-detectors (e.g. BFAST[]), or Bayesian online CPD methods should also be employable as labeling the data into healthy and – after the changepoint – unhealthy parts. Which of these detectors are used (instead of Predict & Compare) should be easily selectable in the implementation of the proposed labelling method.

(*) Assumption about the nature of considered data:

1. The process under observation has cycles.
2. There are features which can be aggregated across each cycle.
3. The resulting time series of aggregated values can be predicted from the values on sub-intervals of the time series.
4. The predictions of the time series' values beyond the sub-intervals may not coincide with the true data.
5. Once a value with unhealthy state is detected, all the values beyond are automatically labelled unhealthy, until the end of the time series.

Remark: If in the feature data at hand no trend can be detected, while the nature of the data changes continuously, then the Predict & Compare method will not work. In this case, a different feature or should be considered.

Example 1: Data with constant mean but first slowly and then strongly varying local variance parameter. In this case the prediction of the trend is always constant, while the changes of the variance may be crucial for the state of health of the monitored process. Here, the health indicating feature should be switched from the mean to the variance.

Example 2: (Experiment 5 from the Used Case): Here, the Predict & Compare method has been used for the case of the simple-most prediction method as the predictor F_t. The respective size of I and J were 60 and 30. The criterion for detection ('compare') was taken to be $M_J < 1.2 \cdot M_I$

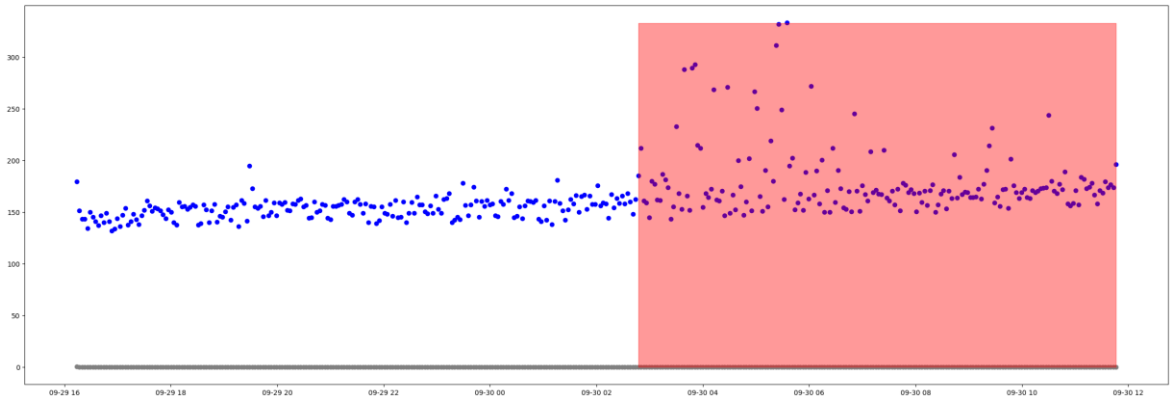

Figure 2: The (single) feature used in this case for health detection was the intra-cycle standard deviation on the respective 'look-back'' window I (calculated at each t). The aggregation method was the mean. The time of the unhealthy state coincides with an increase in local dispersion (standard deviation). The healthy states are labelled '0', the unhealthy ones '1'.

Pseudocode: Labelling into States ('healthy'=0 / 'unhealthy'=1) under Assumption (*)

Input: Monitored time series data $v: \{1, \dots, n\} \to \boldsymbol{R}^p$ with cycle-length m, feature-set $S = \{s_1, \dots, s_p\}$.

1. v is monitored and saved.
2. Each feature in S is aggregated over each cycle of v, resulting in data $w: \{1, \dots, k := \frac{n}{m}\} \to \boldsymbol{R}^p$.
3. For each of the p components of the resulting time series w, a labelling function (e.g., Predict & Compare) is applied, resulting in a time series $z: \{1, \dots, k\} \to \{0, 1\}^p$
4. The time series of aggregated features w and labels z are saved.

Output: Time series of features $w: \{1, \dots, k\} \to \boldsymbol{R}^p$, time series of labels $z: \{1, \dots, k\} \to \{0, 1\}^p$

Pseudocode: Training of Health State Classifier

Input: Time series of features $w: \{1, \dots, k\} \to \boldsymbol{R}^p$, time series of labels $z: \{1, \dots, k\} \to \{0, 1\}^p$.

1. Calculate a univariate label time series $\bar{z}: \{1, \dots, k\} \to \boldsymbol{R}$ from the p-variate label sequence z.
2. Train a supervised classification machine learning model C: using data $\{(w_1, \bar{z}_1), (w_2, \bar{z}_2), \dots, (w_k, \bar{z}_k)\}$ with two states ('0'=Healthy, '1'=Unhealthy).
3. Save the trained classification model.

Output: Classifier (ML model mapping $\{v: \{1, \dots, k\} \to \boldsymbol{R}\}$ to $\{\bar{z}: \{1, \dots, k\} \to \{0, 1\}\}$)

2 Health-Indicator - Classification

The state of health is indicated by one of two states. An algorithm trained on inputs given by feature valued sequences returning 0/1-valued sequences is a supervised classification model.

The health-state classifier is a supervised classification model.

Pseudocode: Training of Health State Classifier

Input: Time series of features $w: \{1, \dots, k\} \to \boldsymbol{R}^p$, time series of labels $z: \{1, \dots, k\} \to \{0, 1\}^p$.

4. Calculate a univariate label time series $\bar{z}: \{1, \dots, k\} \to \{0,1\}$ from the p-variate label sequence z.
5. Train a supervised classification machine learning model C: using data $\{ (w_1, \bar{z}_1), (w_2, \bar{z}_2), \dots, (w_k, \bar{z}_k) \}$ with two states ('0'=Healthy, '1'=Unhealthy).
6. Save the trained classification model.

Output: Classifier (ML model mapping $\{v: \{1, \dots, k\} \to \boldsymbol{R}\}$ to $\{\bar{z}: \{1, \dots, k\} \to \{0, 1\}\}$)

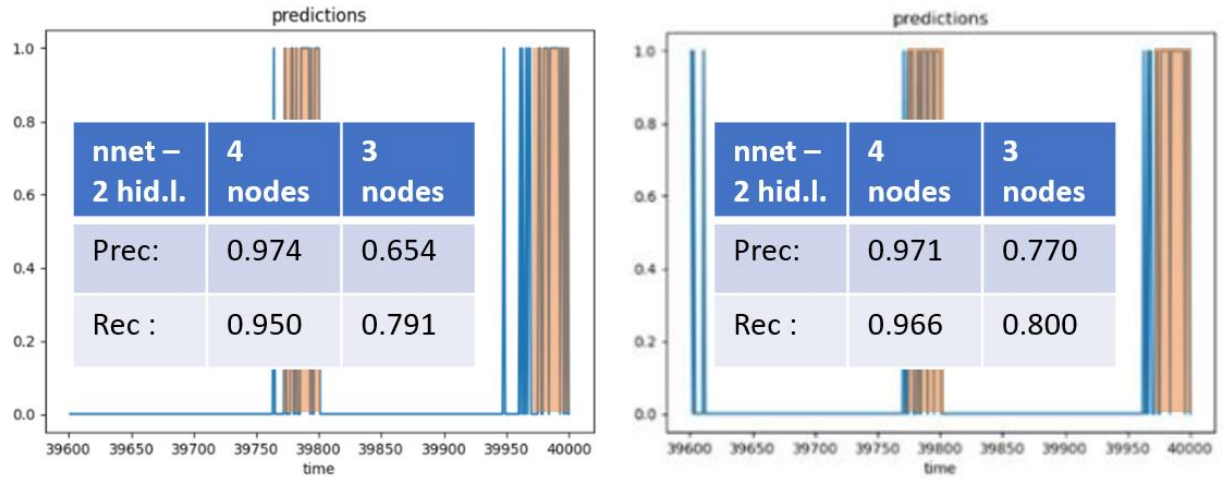

Figure 3: Supervised classifier. Performance is measured with precision and recall. Left: One feature ($p = 1$), Right: Two features ($p = 2$). It is seen that the number of nodes in the second layer has a much higher influence on the performance than the number of included features.

3 Remaining-Useful-Lifetime (RUL) Prediction

A predictive learning model learning real-valued functions is a regression model.

An algorithm trained on sequences of (also real-valued feature values) and times until the first 'unhealthy' time is a supervised regression learning model.

The RUL predictor is a supervised regression learning model.

Pseudocode: Training of RUL Predictor

Input: Time series of features $w: \{1, \dots, k\} \to \boldsymbol{R}^p$, time series of labels $\{\bar{z}: \{1, \dots, k\} \to \{0, 1\}\}$.

1. For each time $s \in \{1, \dots, k\}$, consider the time until the first time t with $\bar{z}(t) = 1$. Call it T_s.
2. Train a supervised regression machine learning model $T: \{1, \dots, k\} \to \boldsymbol{R}_+\}$ using data $\{ (w_1, T_1), (w_2, T_2), \dots, (w_k, T_k) \}$.
3. Save the trained regression model.

Output: Predictor (ML model mapping $\{w: \{1, \dots, k\} \to \boldsymbol{R}^p\}$ to $\{T: \{1, \dots, k\} \to \boldsymbol{R}_+\}$)

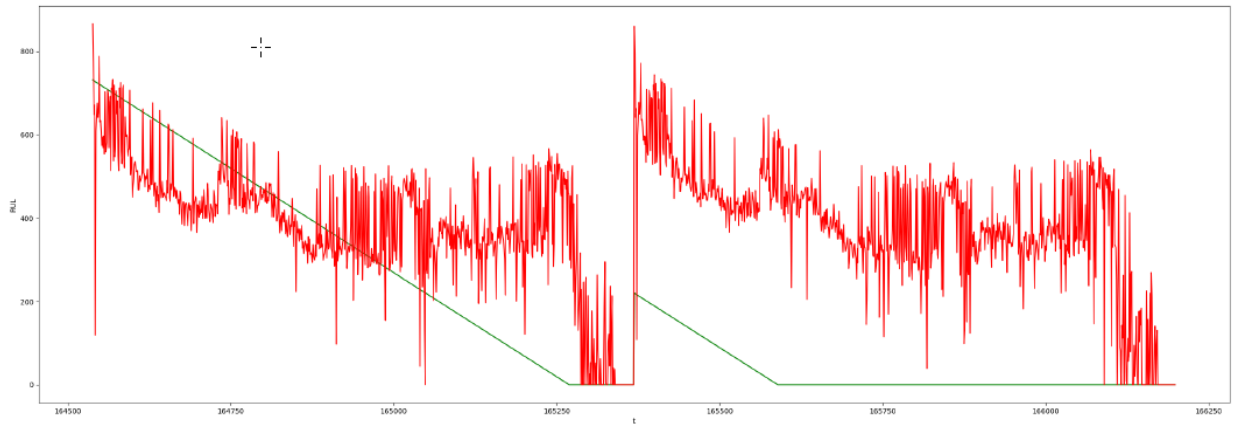

Figure 4: Predicted values (red) of an LSTM-learning model used in the RUL-predictor applied to noisy versions of the same data set (Experiment 5), presented next to ground truth (green). The performance is low, only one feature was used ($p = 1$). In this case, the combined use of the classifier and the RUL-predictor in case of real applications may be essential.

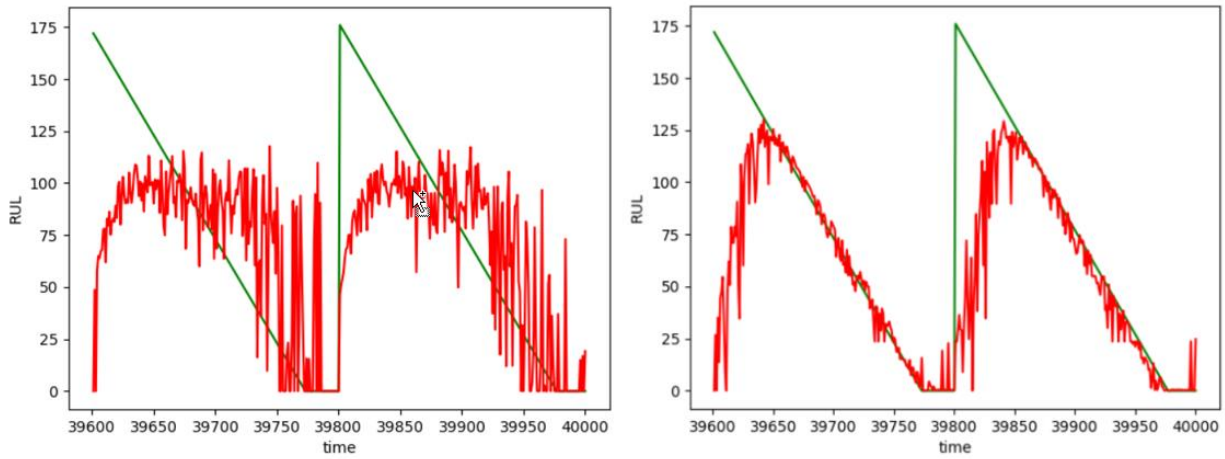

Figure 5: Comparison of performance of a RUL-predictor applied to a synthetic data set (see below). Left: a single feature was used. Right: Here, $p = 2$. Performance - Left: MAD=32.6, Right: MAD=18.8.

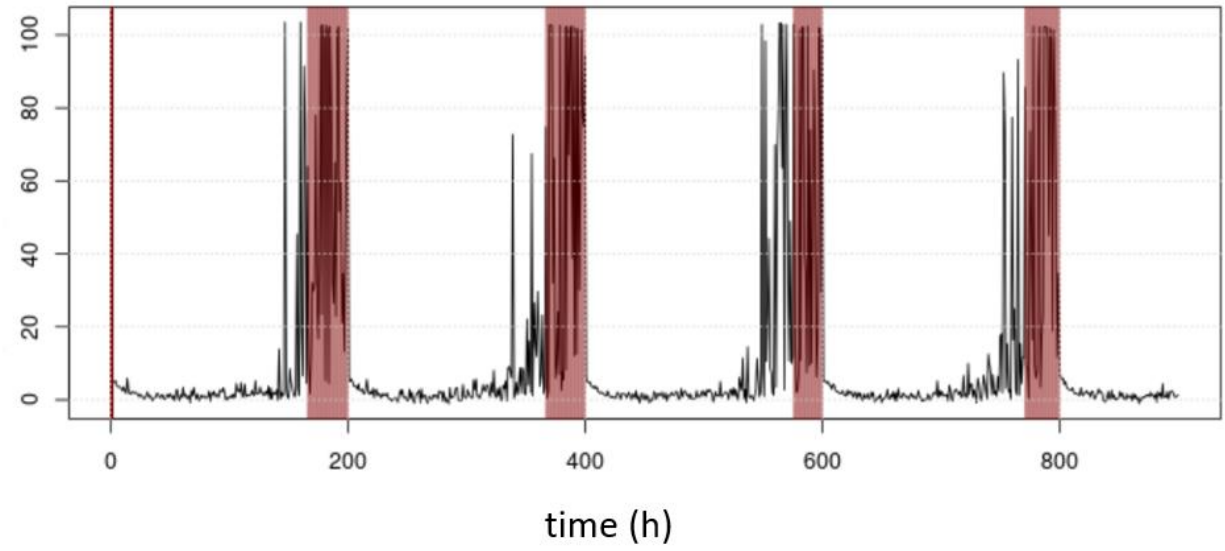

Figure 6: Data used in previous example (Fig. 4), first feature shown. It is seen that in comparison with Fig. 2, that the feature shows the 'unhealthy' state of operation (ground truth: red) in a much more pronounced way. The number of cycles used in the training process was 200 (shown: 4).

Summary

The prototype of a Predictive Maintenance application has been presented. It consists of 5 steps:

1. Defining the Features
2. Labelling the data for training
3. Training the classification model
4. Training the regression model
5. Presenting the results (Estimates of state of health, RUL, and estimated error)

Steps 1 and 5 are tasks which are not part of work-package T1.4.

Step 1 should include the COMPAS' software suite's ability to set thresholds for various aggregated values (mean, max, min, quantiles, ….) to formulate conditions which – for each point in time – are fulfilled, or not (see Fig. 7). This produces new features, which are potentially much better health indicators than the raw sensor values. A set of raw features (from sensors and derived ones) from the rule set should be handed over to Step 2.

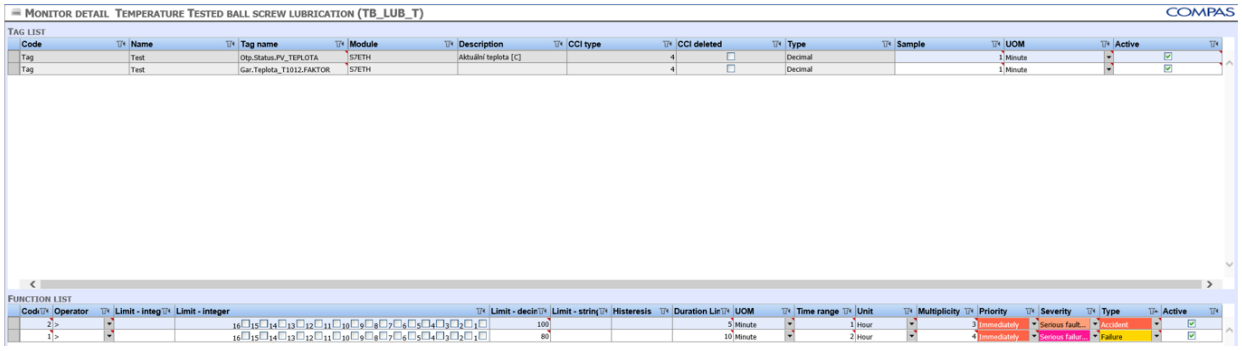

Figure 7: Rule based thresholding system to identify useful derived features based on experience with the practical handling of the production process resulting in typical forms of data streams. These derived features together with the raw sensor values may form the ideal set of health indicating features S.

Step 5 includes receiving the sequences of estimates of $\overline{z:\{1,\dots,k\}\to\{0,1\}}$ and $T:\{1,\dots,k\}\to\boldsymbol{R}_+$ and the performance measures (training error, and if ground truth is available: testing error). Finally, the presentation of these together with the user interface for Step 1 might look like the arrangement sketched in Fig. 8.

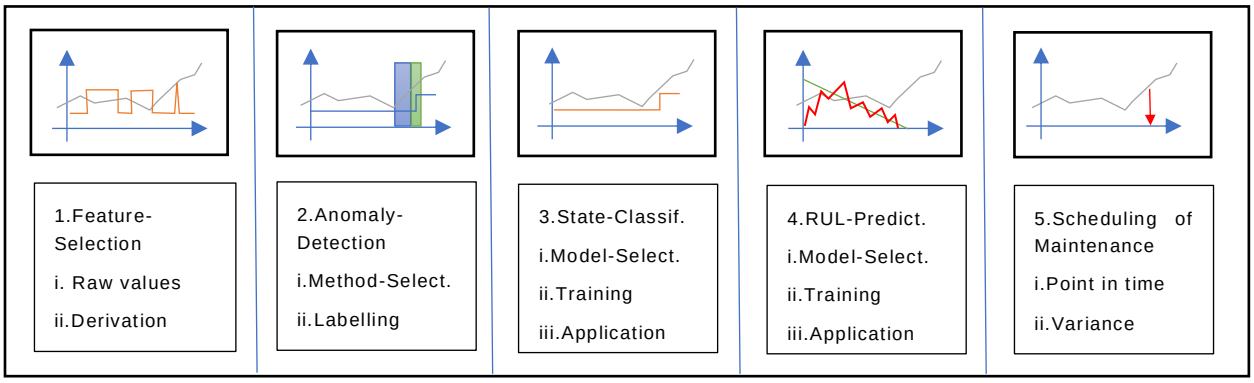

Figure 8: Sketch of a possible arrangement of the five tasks in the PdM software suite.

Appendix: Performance of R-Code:

1. Install R from here: https://cran.r-project.org/bin/windows/base/
2. Call it, and in the window with the > - prompt: type 'source('PMM.R')'.
   PMM.R stands for Predictive Maintenance Module.
3. Follow the indications in the Skript:
   D <- agg(direc='path_to_folder_with_cycle_data/') # Don't forget slash!
   show_D(D) # To see result again.
4. Pick best feature that could be acting as an indicator (probably). For the 'Fifth' experiment, the features 3, 9, 15 are good indicators. This will be called 'feat' in next step:
5. D <- detect_anomaly(D, feat=3, perc_thresh=10)
   With perc_thresh you can control the threshold. It is given at the level for which that many percent of the health indicator feature is above the threshold.
6. The function shows two results, one for the standardized feature, and one for the original scale, together with the resulting health-classification (0/1-valued).
7. This can then be used for determining the multiplicities (COMPAS-software suite) and for training of the classification and RUL-Prognostics part. (Still work-in-progress)

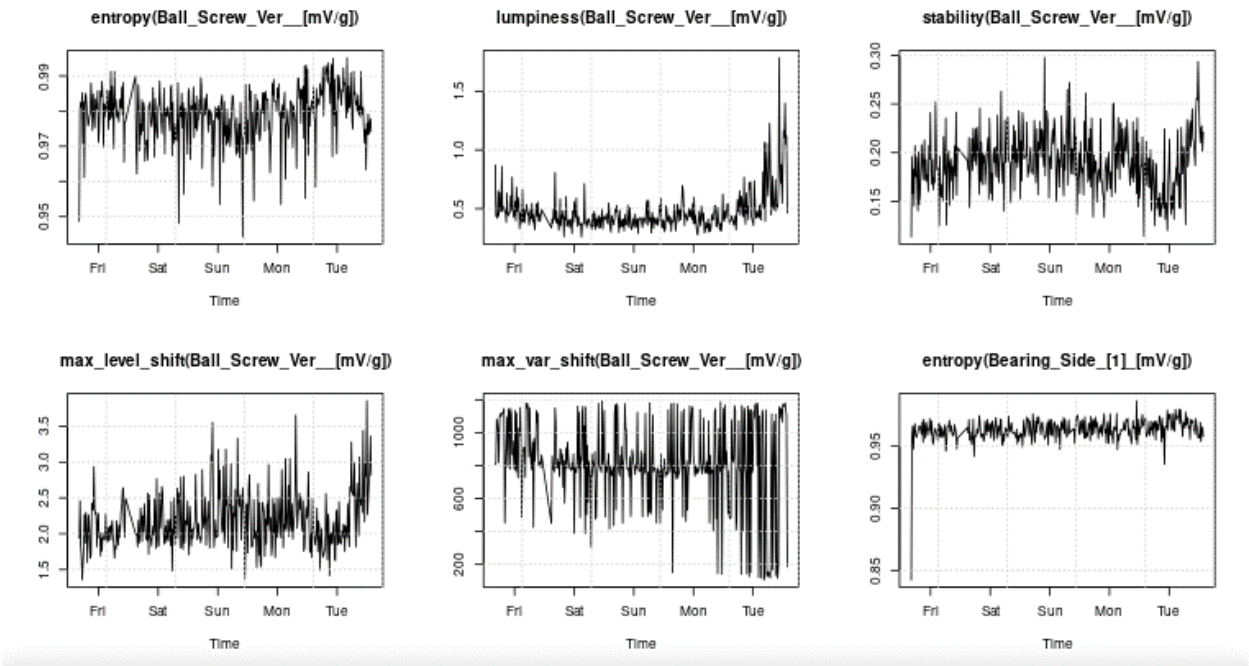

Figure 9: Result of "D = agg(feat, I)" (Step 3) : Here, feat is a vector of names of features from the 'tsfeatures'-library package by Yang and Hyndman, and I is a vector of columns representing the raw features that will be considered Here, I=c(2,3,4,5,6,7). It is seen that feature 'lumpiness' is well suited as a detector of the finally diverging wear process.

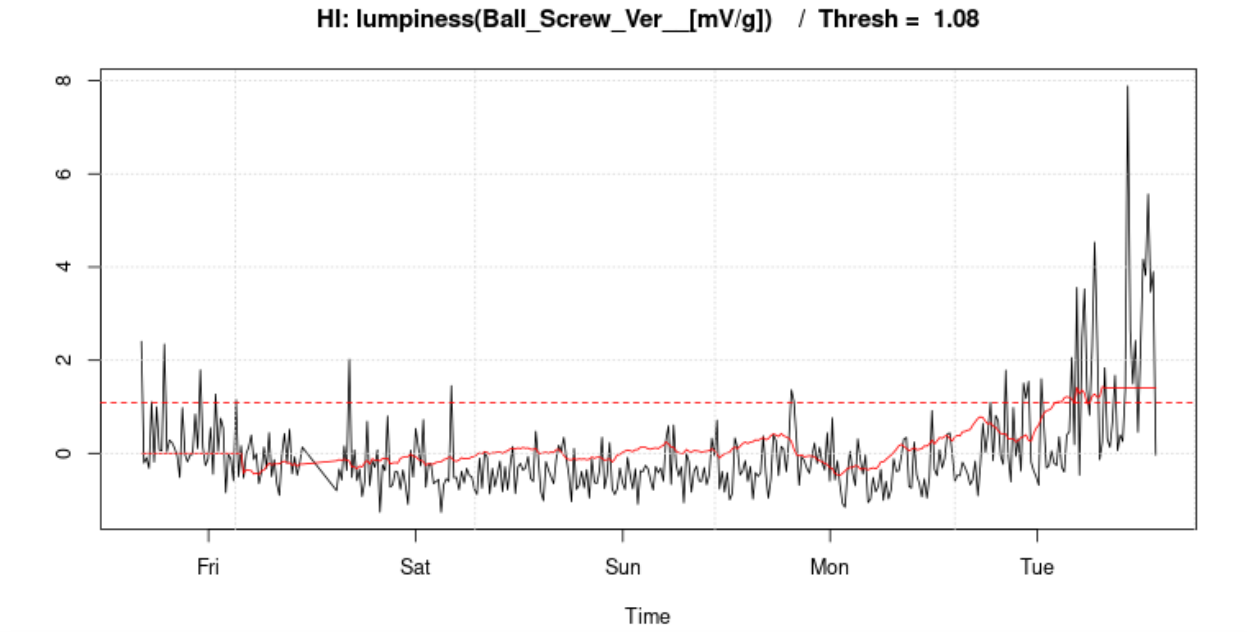

Figure 10: Here, Step 5 is carried out for feature 3: ('D <- detect_anomaly(D, feat=3, perc_thresh=10)'). The red curve is the Predict and Compare anomaly detector. The desired threshold is returned by the condition that it surpasses the threshold at perc_thresh percent of the time.

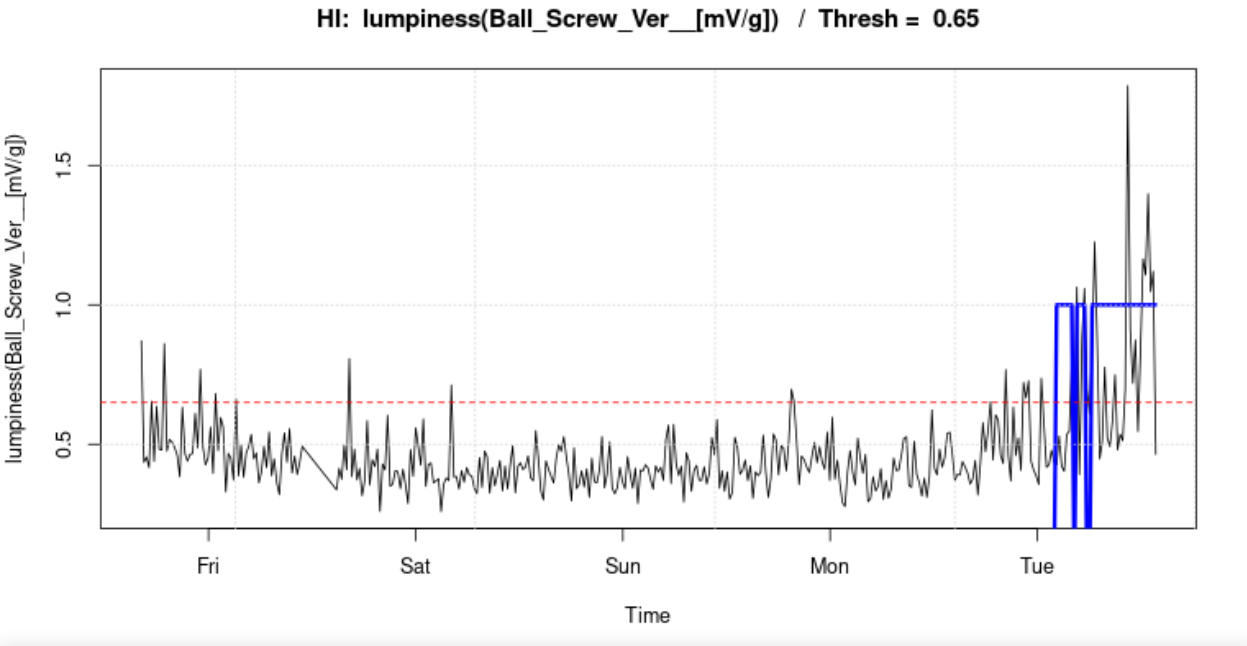

Figure 11: Final Result of the Anomaly Detection – the blue 0/1 classification is the health label. Its intervals of surpassing the threshold (dashed red line) can be used to set the multiplicity parameter of the user-interface of COMPAS' software suite. It can also be used for training of the classifier and RUL predictor.