On the relevance of preprocessing in predictive maintenance for dynamic systems

Carlos Cernuda

Abstract The complexity involved in the process of real-time data-driven monitoring dynamic systems for predicted maintenance is usually huge. With more or less in-depth any data-driven approach is sensitive to data preprocessing, understood as any data treatment prior to the application of the monitoring model, being sometimes crucial for the final development of the employed monitoring technique. The aim of this work is to quantify the sensitiveness of data-driven predictive maintenance models in dynamic systems in an exhaustive way.

We consider a couple of predictive maintenance scenarios, each of them defined by some public available data. For each scenario, we consider its properties and apply several techniques for each of the successive preprocessing steps, e.g. data cleaning, missing values treatment, outlier detection, feature selection, or imbalance compensation. The pretreatment configurations, i.e. sequential combinations of techniques from different preprocessing steps, are considered together with different monitoring approaches, in order to determine the relevance of data preprocessing for predictive maintenance in dynamical systems.

1 Introduction

Nowadays the volume of data is exploding, and the costs of collecting, storing and treating them are affordable for many, making big data solutions more science and less fiction. In this world submerged by a data tsunami, predictive maintenance is not an exception. In fact the advances in cheaper, smaller and much more accurate sensors development, together with highly sophisticated communication protocols, have widely contributed to a continuous rise of data-driven approaches in predictive maintenance.

Carlos Cernuda
BCAM - Basque Center for Applied Mathematics, Bilbao, Spain, e-mail: ccernuda@bcamath.org

In any data-driven application in general, thus for predictive maintenance in particular, preprocessing [132] is of uppermost importance in order to make the data meaningful and usable, driving the path from potential to real information. Depending on the author, preprocessing can take different meanings. Some separate, for instance, data compression approaches, such as feature selection, from preprocessing. We will consider any treatment performed to the data before training a model as *preprocessing*. Then, data cleaning, noise filtering, normalizing, feature selection are part of it, among others.

Therefore, we can think of preprocessing as a step formed by several steps, each or them with a particular purpose, whose order could be sometimes interchanged but in which the commutative property is in general not fulfilled. Considering the amount of possible steps, the variety of possible approaches per step, and the non-commutativity between them, the amount of options explodes existing no guaranty that a combination of preprocessing actions would behave better than no preprocessing the raw data at all [39].

The data involved in each problem related to predictive maintenance have specific properties. For instance, data related to fault detection tend to be highly imbalanced because the information regarding faulty situations is much less frequent than the one regarding fault-free situations. In general, the properties of the data should be taken into account when choosing a preprocessing strategy. Unfortunately the task does not provide enough information, meaning that not all datasets used for a task have the same properties. For example, not all datasets for remain useful life (RUL) prediction problems are the same. The properties of each dataset have to be determined. Moreover, sometimes, with the same properties, a preprocessing scheme works for one problem and not for another. Some general hints are provided in the definitions of the different strategies.

In predictive maintenance accurate models are necessary, but accurate today could become inaccurate tomorrow, making robust long-lasting models also a requirement, specially in highly dynamic systems. Proper preprocessing strategies are the foundation of the construction of a robust accurate model.

The rest of the work is as follows. Section 2 establishes a taxonomy, provides brief but beyond a mere citation descriptions of several techniques for each of the preprocessing steps following the previously provided taxonomy, and presents several modeling techniques meant for system monitoring in predictive maintenance. Section 3 fully describes the datasets that define the different scenarios, the complete experimental setup, as well as the evaluation schemata that would allow for a fair comparison of the proposed pretreatment configurations, and comments about the results achieved. Finally, Section 4 concludes the study.

2 Preprocessing

We define *preprocessing* as the set of actions performed to raw data prior to a subsequent modeling performance, with the aim of improving the modeling capabilities. The improvement could be understood in several ways, such as increasing accuracy, increasing robustness, shortening computational time, decreasing memory and/or computational power requirements, or reducing monetary costs.

The perfect result would be a combination of several of those (usually conflicting) objectives, leading to multi- and many-objective solutions (in which an algorithm is trained in order to find the best preprocessing strategy) that are far beyond the scope of this work. Usually, the objectives are dependent on the problem and the final user requirements. Therefore, we will focus separately on accuracy and robustness, assuming that the methods are fast enough for our requirements as well as affordable in time, technical resources, and money.

2.1 Taxonomy

The taxonomy we are presenting here is an ordered taxonomy, meaning that the steps, if included in our strategy, should be performed in the given order. Since some of those procedures deal with some calculations using the data (e.g. averages), then any transformation made would affect those calculations in the subsequent steps, which could lead to different resulting actions. We first present the six preprocessing steps, and then we develop in detail the most relevant approaches in each one of them.

1. **Data cleansing**. Most data-driven techniques rely on the supposition of complete, reliable noise-free data. But real-world data are not such ideal clean data, being necessary to define strategies to deal with outliers and noise. Moreover, due to the nature of the data or due to a lack of an adequate data acquisition strategy, redundant or irrelevant features could be considered in the dataset, which could be treated both in the data cleansing step or later in the feature engineering step[1]. Despite expert knowledge could be extremely helpful for data cleansing, we assume a lack of it so that we focus on data-based strategies. Some of the parts of the taxonomy are interconnected. For instance, noise treatment is usually attempted through filtering (data transformation) or compression (data engineering), as well as redundancy and irrelevancy, which are usually overcome through data engineering. Therefore, those cases will be treated in their corresponding steps, being the link mentioned.

[1] It is not irrelevant when the treatment happens, because there are several steps between data cleansing and feature engineering that could be very sensitive to redundancies or heavily affected by features that in the end are irrelevant.

2. **Data normalization**. Data coming from diverse heterogeneous origins is collected with ease, which makes actual datasets a compendium of datasets obtained in different parts of the system in different manners. This datasets fusion, known as data integration, is not considered by many authors (including us) as part of preprocessing, but as part of data collection. Some algorithms are highly sensitive to the variety of scales and ranges of the variables, which could lead to a performance degradation if no homogenization is performed in the data.

3. **Data transformation**. Despite the previous steps and some of the posterior ones mean indeed transformations of the data, we reserve this name for transformations in the data by means of certain functions, motivated by knowledge about the system. For instance, if we are performing predictive maintenance of certain industrial machinery by using information about the chemical composition of residual wastes by spectroscopic data (named chemometric multivariate calibration), we can use Beer-Lambert Law to realize that the relationship between the chemical composition and the absorbance spectroscopic data (obtained by a logarithmic transformation) is linear. Therefore, the transformation is beneficial for the posterior use of a linear monitoring technique.

4. **Missing values treatment**. Due to several possible causes some values of certain variables could be missing. A naïve approach is to ignore any sample containing a missing value, but sometimes the amount of samples is small or, in case of imbalanced data, the minority class could become more minor even if we adopt such a destructive approach. The obvious alternative is filling the holes, but how? Depending on the size and intrinsic characteristics of the data, the filling strategy could be tricky.

5. **Feature engineering**. There is not a standard definition of feature engineering. By it, we mean the employment of one or more of: *feature selection* (determination of the most important features according to certain quality criteria), *feature extraction* (creation of new features from some or all the original ones) and *discretization* (transforming continuous features into discrete ones using bins).

6. **Imbalanced data treatment**. If our predictive maintenance problem is supervised so that certain type of samples are extremely rare compare to the others (minority class), then we are facing an imbalanced learning problem. There are two logical ways to proceed: (i) balancing somehow the data, and (ii) compensate giving somehow more importance to the samples from the minority class.

The former is related to *sampling* techniques, and the latter to *weighting* techniques.

2.2 Data cleansing

Data cleansing is a complicated task in which we frequently have to make strong assumptions. Some of those assumptions, might hold theoretically but not in real world data. Therefore, sometime we walk on quicksand. An example we will show right afterwards is the implicit assumption of Gaussian behavior when applying outlier detection based on Mahalanobis distance. As aforementioned, data cleansing deals with several data artifacts, such as outliers, noise, redundancy or irrelevancy.

The detection of *outliers*, understood as feature values that are too far from the general acceptable trend, and the posterior action on those identified outliers is a tricky task. First, how do we identify the *general acceptable trend*?. Second, how do we quantify what *too far* means? Most of the approaches are based on thresholds from distances in certain representation of the feature space.

We will consider two important approaches, which relevance comes not only because they are widely used but also because they can be updated incrementally for data streams. They are based on Mahalanobis distance [77], and on chi-square approximations of the orthogonal (Q) and score (T^2) distances from principal components analysis (PCA) [61]. As it is indeed an orthogonal transformation, PCA will be briefly described in Section 2.4.

2.2.1 Outlier detection based on Mahalanobis distance

Mahalanobis distance [77] is defined for two vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ as

$$d_M(\mathbf{x}_i,\mathbf{x}_j) = \sqrt{(\mathbf{x}_i-\mathbf{x}_j)^T S^{-1}(\mathbf{x}_i-\mathbf{x}_j)} \tag{1}$$

It takes into account the covariance matrix S, where S_{ij} is the covariance between $\mathbf{x}_i$ and $\mathbf{x}_j$ and S_{ii} is the variance of $\mathbf{x}_i$. Thus we are considering elliptic regions, instead of circular ones, of equidistant points. Figure 1 shows a 2-D example where the point marked with the red square would not be considered as an outlier according to Euclidean distance, but it would be in terms of Mahalanobis distance, which seems to be more reasonable.

The outlier identification procedure consists in calculating the Mahalanobis distance from each sample to a central point and check whether it exceed certain threshold. The mean is the classical central measure, but it is not robust against outliers. Also the covariance matrix is not a robust dispersion measure. The robustness can be

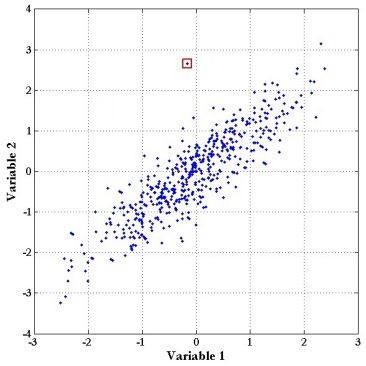


Fig. 1 Example of outlier according to Mahalanobis distance, that would not be so according to Euclidean. Considering Euclidean distance, the lines of points with a constant distance to a central point form a circumference. But considering Mahalanobis distance, the shape of those line is elliptical, and adapted to the overall shape of the cloud of points.

assumed if the number of samples is quite big, that is usually the case in chemometrics. Robust alternatives to the mean and the covariance matrix are, respectively, the *robust location estimator* and the *minimum covariance determinant*, which are the mean and covariance matrix of a subset of the original data set. For further information see [116].

If we denote by $\mathbf{x}_c$ the chosen center and by S_c the chosen dispersion matrix, then Mahalanobis distance from a sample $\mathbf{x}_i$ to the center is given by

$$d_M(\mathbf{x}_i) = \sqrt{(\mathbf{x}_i - \mathbf{x}_c)^T S_c^{-1} (\mathbf{x}_i - \mathbf{x}_c)} \tag{2}$$

Assuming that the multivariate data follows a multivariate normal distribution, then the squared Mahalanobis distance follows a χ^2_N distribution, with N the number of variables. Then a sample would be considered as an outlier if its distance to the mean is higher than the threshold given by a α quantile, $\chi^2_{N,\alpha}$.

For the incremental case, we just need to be able to incrementally update the inverse of the covariance matrix, which is defined as

$$\Sigma_N = \frac{1}{N} \sum_{i=1}^{N} (x_i - \overline{X}_N) \cdot (x_i - \overline{X}_N)^T \tag{3}$$

Then, for the extended data stream considering an extension with one single sample,

$$\Sigma_{N+1} = \frac{1}{N+1} \sum_{i=1}^{N+1} (x_i - \overline{X}_{N+1}) \cdot (x_i - \overline{X}_{N+1})^T$$

If we split the sum in two parts, from 1 to N and $N+1$, we get

$$\Sigma_{N+1} = \frac{1}{N+M}\sum_{i=1}^{N}(x_i - \overline{X}_{N+1})\cdot(x_i - \overline{X}_{N+1})^T + \frac{1}{N+1}(x_i - \overline{X}_{N+1})\cdot(x_i - \overline{X}_{N+1})^T$$

We denote both addends as A_1 and A_2 respectively, and expand them separately.

Firstly,

$$A_1 = \frac{1}{N+1}\sum_{i=1}^{N}(x_i - \overline{X}_{N+1})\cdot(x_i - \overline{X}_{N+1})^T$$

Taking into account that the incremental update of the mean is given by

$$\overline{X}_{N+1} = \frac{N\overline{X}_N + x_{N+1}}{N+1} \tag{4}$$

Then,

$$-\overline{X}_{N+1} = -\overline{X}_N - \frac{1}{N+1}(x_{N+1} - \overline{X}_N)$$

Denoting $\overline{C} := x_{N+1} - \overline{X}_N$, and substituting,

$$A_1 = \frac{1}{N+1}\sum_{i=1}^{N}\left(x_i - \overline{X}_N - \frac{1}{N+1}\overline{C}\right)\cdot\left(x_i - \overline{X}_N - \frac{1}{N+1}\overline{C}\right)^T$$

As $(A-B)\cdot(A-B)^T = A\cdot A^T - A\cdot B^T - B\cdot A^T + B\cdot B^T$, and $\overline{C}$ is constant, then

$$\begin{aligned}
A_1 = &\frac{1}{N+1}\sum_{i=1}^{N}\left(x_i - \overline{X}_N\right)\cdot\left(x_i - \overline{X}_N\right)^T - \\
&- \frac{1}{(N+1)^2}\sum_{i=1}^{N}\left(x_i - \overline{X}_N\right)\cdot\overline{C}^T - \\
&- \frac{1}{(N+1)^2}\sum_{i=1}^{N}\overline{C}\cdot\left(x_i - \overline{X}_N\right)^T + \\
&+ \frac{1}{(N+1)^3}\sum_{i=1}^{N}\overline{C}\cdot\overline{C}^T = \\
&= \frac{N}{N+1}\Sigma_N + \frac{N}{(N+1)^3}\overline{C}\cdot\overline{C}^T
\end{aligned}$$

Secondly,

$$A_2 = \frac{1}{N+1}(x_{N+1} - \overline{X}_{N+1})\cdot(x_{N+1} - \overline{X}_{N+1})^T$$

From Equation (4), we know that

$$-\overline{X}_{N+1} = -x_{N+1} + \frac{N}{N+1}(x_{N+1} - \overline{X}_N)$$

Therefore,

$$A_2 = \frac{N^2}{(N+1)^3}\overline{C}\cdot\overline{C}^T$$

Then, since $\overline{C} := x_{N+1} - \overline{X}_N$,

$$\Sigma_{N+1} = \frac{N}{N+1}\Sigma_N + \frac{N}{(N+1)^2}(x_{N+1}-\overline{X}_N)\cdot(x_{N+1}-\overline{X}_N)^T \tag{5}$$

In order to obtain the inverse of the covariance matrix, one option is to update the covariance matrix and calculate the inverse. This requires a huge computational effort unless the number of variables is very low, which is not usually the case. Therefore, a direct update of the inverse covariance matrix is preferable.

The properties of the matrices involved in Equation (5) allow us to compute the new inverse as a perturbation of the old one by using the following Lemma [1].

Lemma 1 (General Sherman-Morrison formula). *Suppose $A \in \mathscr{M}_n$ is an invertible matrix, and v and w are vectors of length n so that $1 + w^T A^{-1} v \neq 0$. Then,*

$$\left(A + v\cdot w^T\right)^{-1} = A^{-1} - \frac{A^{-1}v\cdot w^T A^{-1}}{1 + w^T A^{-1} v} \tag{6}$$

where $v \cdot w^T$ is the outer product of v and w.

If we identify $A := \frac{N}{N+1}\Sigma_N$, $v := \frac{N}{(N+1)^2}(x_{N+1}-\overline{X}_N)$, and $w := x_{N+1}-\overline{X}_N$, then we have

$$1 + w^T A^{-1} v = 1 + \frac{1}{N+1}(x_{N+1}-\overline{X}_N)^T\Sigma_N^{-1}(x_{N+1}-\overline{X}_N)$$

that is never null because Σ_N is positive semi-definite, then so its inverse.

In Equation (5), inverting both sides

$$\Sigma_{N+M}^{-1} = \left(\frac{N}{N+1}\Sigma_N + \frac{N}{(N+1)^2}(x_{N+1}-\overline{X}_N)\cdot(x_{N+1}-\overline{X}_N)^T\right)^{-1} \tag{7}$$

that corresponds to the left part of (6) in the Lemma, with the previous identifications of A, v, and w.

By Sherman-Morrison formula,

$$\Sigma_{N+1}^{-1} = \frac{N+1}{N}\Sigma_N^{-1} - \frac{\frac{N+1}{N}\Sigma_N^{-1}\cdot\frac{N}{(N+1)^2}(x_{N+1}-\overline{X}_N)\cdot(x_{N+1}-\overline{X}_N)^T\cdot\frac{N+1}{N}\Sigma_N^{-1}}{1 + (x_{N+1}-\overline{X}_N)^T\cdot\frac{N+1}{N}\Sigma_N^{-1}\cdot\frac{N}{(N+1)^2}(x_{N+1}-\overline{X}_N)} \tag{8}$$

Therefore, taking common factor $\frac{N+1}{N}$, we get

$$\Sigma_{N+1}^{-1} = \frac{N+1}{N} \cdot \left(\Sigma_N^{-1} - \frac{\Sigma_N^{-1}(x_{N+1} - \overline{X}_N) \cdot (x_{N+1} - \overline{X}_N)^T \Sigma_N^{-1}}{(N+1) + (x_{N+1} - \overline{X}_N)^T \Sigma_N^{-1} (x_{N+1} - \overline{X}_N)} \right) \quad (9)$$

Now, taking the square in equation 2, the square Mahalanobis distance from a sample x_i to the center x_c is

$$d_M^2(x_i) = (x_i - x_c)^T S_c^{-1} (x_i - x_c)$$

Supposed we have prefixed a confidence level α, the threshold for the outlier detection is $\chi_{m,\alpha}^2$, where m is the number of variables. Thus it is independent of the number of samples and, then, fixed during the whole on-line process.

Let us suppose that we have the updated center and inverse dispersion matrix at a time t. Once the next sample, x_{t+1}, from the data stream arrives, its $d_M^2(x_{t+1})$ value is calculated in order to decide whether it is an outlier or not, according to the current center and inverse dispersion matrix. If it is not an outlier, the previous center (obtained by averaging) and inverse dispersion matrix can be incrementally updated as shown.

2.2.2 Outlier detection based on χ^2 approximations of Q and T^2 statistics

Suppose that we have a centered data matrix $X \in \mathscr{M}_{M,N}$ where the columns correspond the predictor variables. Therefore we can consider that we are working in a N-dimensional space E. Once selected a number a of principal components, principal components analysis algorithm projects the data onto an a-dimensional subspace V, defined by the a first principal components. Then we can consider the orthogonal supplementary subspace of V, $U = V^{\perp}$, that is a $(N-a)$-dimensional, meaning that $V \oplus U = E$. Consequently, any element x in E has unique projections in both V and U so that their sum equals x. The selection of a is crucial for the final result. Nevertheless, the way to determine it is out of the scope of this section, and has been widely treated in the literature.

We are interested in two distance measures: (i) the Mahalanobis distance from the projection of x onto V to the center of the cloud of projections of all the data onto V, called *score distance*, and (ii) the Euclidean distance from x to V, called *orthogonal distance*, which is related with the Euclidean distance to U. Figure 2 provides the geometric interpretation of both statistics for an original three-dimensional data example projected onto a two-dimensional subspace.

The former distance indicates the variation of each sample within the model. It is also known as Hotelling's T^2 statistic, and can be calculated as [76]

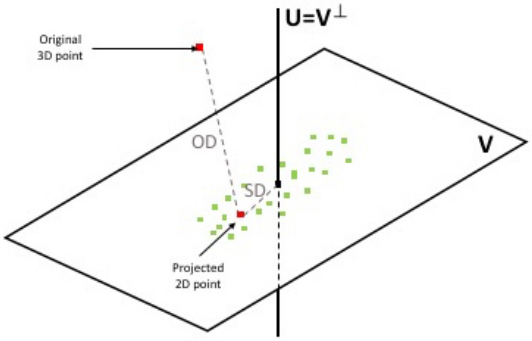


Fig. 2 Geometric interpretation of the score distance *SD* and orthogonal distance *OD* for a three-dimensional example projected onto a two-dimensional subspace V. For visual purposes, we have shown an original 3D point with a huge *OD*. Due to the way the PCs are selected, this is not usually the case, and, unless the point is an outlier, the *OD* is commonly small.

$$T_i^2 = x_i P_a \Lambda^{-1} P_a^T x_i^T = \sum_{j=1}^{a} \frac{t_{ij}^2}{\lambda_j} \tag{10}$$

where $\Lambda = \{\lambda_j\}_{j=1}^{a}$ is a diagonal matrix containing the biggest a eigenvalues and P_a is the loadings matrix.

For a fixed number a of principal components, on the basis of the fact that the data are centered, we can model the score distance, since all random variables t_{ia} have null expectation and variance λ_a/M, as [7]

$$DoF \cdot \frac{T^2}{\overline{T^2}} \sim \chi^2(DoF) \tag{11}$$

where DoF and $\overline{T^2}$ are the *degrees of freedom* and the average Hotelling's statistic respectively. DoF could be estimated by

$$\widehat{DoF} = \frac{2\overline{T^2}^2}{S_{T^2}} \tag{12}$$

where S_{T^2} is an estimation of the standard deviation of T^2. A robust option, based on the interquartile range (IQR) is obtained by solving wrt $\widehat{DoF}$ the equation

$$\frac{1}{\widehat{DoF}} \left[\chi^{-2}(\widehat{DoF}, 0.75) - \chi^{-2}(\widehat{DoF}, 0.25) \right] = \frac{1}{\overline{T^2}} IQR(T_1^2, \ldots, T_M^2) \tag{13}$$

The latter distance, also known as Q statistic, indicates how well each sample conforms to the model, and it can be defined for a given sample x_i as

$$Q_i = \sum_{j=a+1}^{k} t_{ij}^2 \tag{14}$$

where $(t_{i1},\ldots,t_{iN})$ is the i-th row of T, and k is the rank of X.

A similar formula to Equation (11) can be proposed

$$C \cdot \frac{Q}{\overline{Q}} \sim \chi^2(C) \tag{15}$$

It depends only in one parameter C, that can be estimated in an analogous way as in Equation (12).

Now that we have totally determined the distributions of both distances in terms of χ^2 distributions, p-values can be calculated, for a certain chosen critical level α, which are the probability of occurrence of each T^2 and Q. Considering c_i as any of T_i^2 or Q_i, the corresponding p-value is

$$P(c_i) = 1 - [1 - CDF(c_i)]^M \tag{16}$$

where CDF is the cumulative distribution function of the corresponding distribution. If any of the p-values is below the fixed critical level, then the corresponding input is considered as an outlier.

Assuming we keep the principal components fixed, at a time t we can suppose that we have the updated estimated distributions for Q and T^2. Once the next sample, x_{t+1}, from the data stream arrives, its Q_{t+1} and T^2_{t+1} values are calculated in order to decide whether it is an outlier or not, according to the current estimations of the distributions of Q and T^2. If it is not an outlier, the mean values for Q and T^2 can be incrementally updated.

Besides, the new estimations of C and DoF can be done just by incrementally estimate the updated IQR. The calculation of the real IQR requires to store all data in memory. Nevertheless, the estimation could be done based on a window [38, 89, 82] (requiring memory for the samples in the window only), or based on quantile approximations [114]. All this allows us to incrementally extend the outlier detection based on Q and T^2 to data streams.

2.3 Data normalization

Assuming that preprocessing is a preliminary task prior to a subsequent modeling phase using certain method, it is important to understand the characteristics of that method in order to perform a proper data preprocessing.

The most used technique is *mean centering*, consisting on subtract the mean value of every feature (thus column-wise). Some methods, like principal components regression (PCR) or partial least squares (PLS) have connections to distances to a

central location of the distribution of the data. Therefore, if the data is not centered they suffer from certain bias due to the distance to the origin of the raw data points.

Another fundamental normalization technique is *standardization*. Standardization comes from the transformation of a general Gaussian distribution into a standard Gaussian distribution (with null expectation and unitary variance), obtained by mean centering plus dividing column-wise by the standard deviation of every feature. By standardizing we make our data centered and unitary spread, thus correcting differences in the scales and ranges of the features. When employing any monitoring algorithm in which distance calculations are somehow involved, standardization is recommended unless the nature of the features is similar. In such cases, the differences in the ranges of the features is relevant for the process we are monitoring. An example of an algorithm involving distances is support vector machines, in which the widths (distances) between the data groups determined by the support vectors is maximized.

The third normalization approach we will consider is *scaling*. The motivation behind is gaining robustness to tiny feature variances, as well as to avoid zero entries in case of sparse data. In scaling we choose an interval and our data will be scaled so that it fits into that interval. The usual intervals are $[0,1]$, obtained by subtracting the minimum value and dividing by the range, and $[-1,1]$, obtained by dividing mean centered data by the value with largest absolute value in each of the features. The latter is the preferred one for sparse data. Both approaches are highly sensitive in presence of outliers, thus either a proper outlier detection strategy or the use of robust alternatives to the range and standard deviation are recommended.

2.4 Data transformation

The versatility of the data employed in predictive maintenance opens plenty of possibilities when it comes to transformations. There are two main branches in data transformation for predictive maintenance, which we identify as *statistical transformations*, and *signal processing*.

2.4.1 Statistical transformations

The *statistical transformations* are inspired in those transformations historically used in statistical inference. The use of one or another type depends on the application and the type of data.

In Statistics, data transformations are applied when some prior information motivating them is available. Some of the most famous ones are *logit transformation*, from logistic regression and being related to neural networks and deep learning methods,

square root transformation, from quadratic regression, and *reciprocal transformation*, obtaining similar scaling transformations as logit but also applicable to negative values. In general, all those transformations can be generalized by means of the *power transformation* [49], that depends on a parameter λ, being all the aforementioned particular cases for certain λs. As the *identity* is also a particular case, it is possible to infer the most adequate transformation for some given data (by optimizing λ) including not transforming at all (identity). This technique is known as *Box-Cox* [5, 95]. Box-Cox has been successfully employed in fault detection [100].

Another family of transformations with statistical background are the projection on latent subspaces, like PCA and partial least squares (PLS). PCA is easily understandable if we approach it as an iterative procedure. Assuming we have centered data, the first PC will be the single direction on which the variance of the projection of the data is maximum. This direction is always obtainable as a linear combination of all the original features. Once fixed the first principal component PC_1, we consider the orthogonal supplementary subspace of the subspace defined by PC_1 that is a line. As an example, in 3D the orthogonal supplementary subspace of a line is the plane that is orthogonal to it. In Figure 2 the plane V is the orthogonal supplementary subspace of the line U. In this supplementary subspace we can also look for the single direction on which the variance of the projection of the data is maximum, getting PC_2. Notice that, as any direction in the subspace, PC_2 is orthogonal to PC_1. As each supplementary subspace we obtain has one dimension less than the previous one, we can continue with the same process until we end up with one last single line, that is the last principal component (PC_N if we had originally N features). Also in Figure 2, V would be the plane defined by PC_1 and PC_2 (where PC_1 and PC_2 have respectively the direction of the large and small axes of the ellipse formed by the green points), and U would be the line defined by the last component $PC_3 = PC_N$.

PLS could be seen as a supervised equivalent to PCA. It becomes clear when we point out that the procedure for the calculation of the components in PLS (called *latent variables*) is similar to the case of PCA, but the objective is to maximize variance of the projection plus correlation with the target simultaneously. There is also a relevant difference from the algebraic point of view. In PCA, the supplementary space considered is the orthogonal one. Nevertheless, aiming for some flexibility required by the double objective of maximizing not only the variance of the projection but also the correlation with the target, PLS considers a supplementary subspace not necessarily orthogonal. The need of the target makes PLS unfeasible for online outlier detection. The application of a PCA variant is usually referred as performing an orthogonal transformation [101].

Both PCA and PLS are linear transformations, unless we opt for one of their multiple non-linear extensions. There are several recent non-linear transformations that are meant for exploiting the relations among the features. By relevance and usage, the most important ones are locally linear embedding (LLE) [91], isomap [110] and derivatives. They rely on the transformation of the original set of features into a

smaller amount of projections taking into consideration the geometrical properties of clusters formed by instances, or patches of the underlying manifolds. Therefore, this methods could also fit into Section 2.6, because they could be understood as dimensionality reduction approaches.

2.4.2 Signal processing

The heterogeneity in the properties of the data samples, also called *signals*, leaves margin for transformations coming from many sources. We have seen statistical transformations, but they also could arise from Mathematics, Physics or Computer Science. It is a matter of semantics, but usually the word signal is reserved for certain type of data that can be ordered in time. Concretely we will focus on waveform data, because most of the predictive maintenance data are based upon this type. Waveform data can be observed from two related domains: time domain and frequency domain, being possible to move from one to the other and back. Depending on the domain we will distinguish three types of techniques in signal processing [109], which are (i) *time domain*, (ii) *frequency domain*, and (iii) *time-frequency domain*.

The analysis of the *time domain* is the analysis of the original waveform data, which is, from a mathematical point of view, a chronological sequence of the value of certain random variable, having certain expectation, variance, skewness and kurtosis which calculation could be part of the analysis, helping to characterize the signals. An example of time domain analysis is time series analysis [23], being *autoregressive* models one of the most employed ones. By autorregressive we understand that the feature values depend linearly of the previous ones[94, 99], so we would be assuming independence between features. If we think that it is not the case, then fractal time series take into account dependencies between two waveforms in different ways, such as local or global self-similarity, or short-range or long-range dependency [70].

The consideration of the *frequency domain* has several motivations. One of them is the fact that noise is usually affecting our signals, being recommended to use denoiser filters. This filters, when applied in the time domain have huge computational costs, as they imply the application of convolution operations. Meanwhile, in the frequency domain they are just multiplications, as they transform differential equations into algebraic ones. Therefore, it is computationally cheaper to transform the data into the frequency domain, apply a filter there, and transform the filtered data back to the time domain in order to perform any posterior analysis there. There are many possible filters to be applied, even designed, depending on the components of the data we need to filter out [109]. Just as an example, a famous digital filter for smoothing the data is Savitzky-Golay filter [96, 81, 84], which is based on a local low-order polynomial interpolation using for each point a window containing some of its neighbor points. Some filters are also suitable for incremental on-line applica-

tion on a streaming context [102].

The use of signals for modeling the state of a real dynamic systems needs indeed information available in both the time and the frequency domain. For this reason, it is common to use both domains at the same time, moving from one to the other on demand. This use is called *time-frequency domain* analysis.

There are several ways to transform time domain signals into frequency domain signals [8]. We highlight (i) the Fourier transform [120], (ii) The Laplace transform, and (iii) the Z transform [109] (known as the discrete version of the Laplace transform), since they are the most relevant ones. There are efficient algorithms to calculate them as well as their inverses. For instance the fast Fourier transform (FFT) [122, 29, 35] is an efficient algorithm for calculating the Fourier transform. It suffers from a problem because it considers the whole signal. If we are facing, for instance, a fault detection problem trying to identify faults by changes in the signal, we could miss true faults (camouflaged as noise) unless the changes are significantly big wrt the whole signal. A way to overcome this effect is considering the short-time Fourier transform (STFT), that considers a fixed-width time window [109, 25].

Nevertheless there is another issue with STFT, coming from the fact that a good resolution in one domain implies a bad resolution in the other. This force us to choose the width of the window so that there is a fine trade-off between the resolution in both domains. Another solution consists in employing a wavelet transformation, which provides us with the same effect as having dynamic resolutions in time and frequency. There are continuous and discrete wavelet transformations [26, 78], being the latter more computationally efficient.

Some more sophisticated newer approaches were developed afterwards. The *Hilbert Huang transformation* [56], a two-steps method consisting on (i) *empirical mode decomposition*, i.e. the decomposition of the signal into a finite number of intrinsic mode functions, and (ii) *Hilbert transform* of the intrinsic mode functions. The fact that those functions are orthogonal [104], implies that they can be understood as having physical meaning, thus being applicable in predictive maintenance[126].

Finally, the *Wigner Ville distribution* [24] was adapted by Ville [118] from Wigner's work in the field of quantum mechanics. It is a quadratic integral transformation in the form of a two-dimensional Fourier transform of a time-frequency autocorrelation function related to both time and frequency. It is not a window-based method, and it provides with the best resolution. Nevertheless, when a signal is a composition of two signals, there appear cross terms that could interfere (by distortion) the result of the analysis[63]. Otherwise, the study of the differencies in the cross terms could be used in predictive maintenance problems [119].

2.5 Missing values treatment

The appearance of missing values is common in real world data collected remotely and sent synchronously to a central database. In the same way as in the case of outliers, an obvious approach is to ignore samples in which one of more features present a hole. As discussed in that case, sometimes we cannot afford ignoring data. Then there is an obvious alternative, *missing value imputation*. The *what* is obvious, but the *how* is really hard.

Naïve logical options, such as imputing the mean, or median as robust alternative, in numerical features, or the mode, in categorical ones, could be very risky. For instance, in imbalanced data situations the minority class gets great importance in the modeling, thus erroneous imputation could significantly influence model behavior.

In case we have a methodology to compare different samples and check whether they are similar or not just by looking at a subset of the features that defines them, then we could compare a sample with a hole with the samples without holes, and choose for the imputation the value of the hole-free sample. There are alternatives in which the sample to be used as imputer, such as systematically use one sample (*cold deck*) or randomly select from a pool of candidates (*hot deck*). As an example, if the methodology is based on distances using all features and the most similar one (i.e. the closest), then the approach is the same as K-nearest neighbors with $K = 1$. The main drawback of this approach is the difficulty of finding a proper way to compare samples. For instance, if we employ distances and the amount of features is big we would suffer from the *curse of dimensionality* effect [85], being all distances huge and comparable in terms of magnitude.

As an alternative, an option could be a *double-model* strategy, in which a model is created using the fully available hole-free data in order to be used for imputation only. The estimated value could be directly used (*regression imputation*), or it could be slightly modified by adding a random residual (*stochastic regression imputation*). Once the missing values have been imputed, the main model is trained. There are several options depending on how to consider the imputed samples. Some approaches consider them as regular legitimate data samples, and some others underweight them, making them less influential in the main model. In case of computational and/or time expensive models, the imputing model employed is different from the main one, such that it is cheaper and/or faster. For instance, common algorithms used for this purpose are K-nearest neighbors [113], fuzzy K-means [69], Bayesian PCA [83], and multiple imputations by chained equations (MICE) [92].

In special cases in which the features are related, we could use *extrapolation* or *interpolation* methods for imputation. For instance, in data coming from a spectrometer, the different features consist of measurements made at different but close sequence of wavelengths, thus features in close wavelengths should present similar values.

2.6 Data engineering

In data engineering we include three approaches (feature selection, feature extraction, and discretization) that could be performed alone or combined. Nevertheless it is not usual to combine them because they actually result in severely modified data as they are deeply invasive procedures.

2.6.1 Feature selection

By *feature selection* we mean feature subset selection. Some authors consider both concepts as different because there are approaches in which the output is a ranking of all features instead of a subset of them. Nevertheless, we will say just feature selection since the common action is to use rankings to get a subset by truncation.

Feature selection can be understood as an optimization process in which the aim is to find a collection of features that makes certain quality criteria optimum. The simplest approaches, in which one single criterion is optimized, e.g. minimizing the root mean squared error (RMSE) of prediction in a regression problem, can be considered as single-objective optimization problems in which the objective function to be optimized is the quality criterion.

There are two types of feature selection (FS) approaches [44]:

- **Filters** [93]. They ignore the posterior task and focus only on the characteristics of the data to perform the selection i.e. the criteria to be optimized is intrinsic to the data (e.g. mutual information between the features and the target). They could be understood as some kind of preprocessing selection. They are fast, but they usually ignore the possible redundancy in the data because most of the approaches evaluate the features independently from each others.
- **Wrappers** [65]. The modeling task (e.g. classification or regression) is understood as a black-box, whose performance using the subset of selected features is the goodness of the selection (performance optimization). They can deal with the redundancy, but they are usually computationally expensive, and they tend to overfit if the amount of available data is not enough.

Some taxonomies include a third type, *embedded methods*, that are those methods in which the selection is internal to the model. As then the feature selection cannot be decoupled from the training, we cannot consider them as preprocessing, thus we keep the 2-types taxonomy.

As filter methods rely on the characteristics of the data, the most renowned methods are based on statistical measures suitable for establishing dependencies and/or relationships between inputs and outputs, e.g. sensors information and machinery condition. Perhaps the most important filter method is *correlation-based feature selection* [46], in which the correlation between the features and the target is used.

There are plenty of ways, some employing problem-specific information, for defining what we understand by correlation, leading to different versions of the algorithm. Any way, specific to the predictive maintenance task, to establish a quantifiable relation between a feature (or a subset of them) and the output of the task that is capable of comparing/ordering different features (resp. subsets) could be used as a measure of correlation.

Recently, Brown [11] has found a generalization framework of some of the most extended families of filter methods that facilitates their understanding, given by

$$J_{Brown} = I(X_n;Y) - \beta \sum_{k=1}^{n-1} I(X_n;X_k) + \gamma \sum_{k=1}^{n-1} I(X_n;X_k|Y) \tag{17}$$

where n is the number of features, X_i the i-th feature, Y the output, and $I(X;Y)$ is the mutual information shared by X and Y [103].

The approaches subsumed in the framework, just by playing with β and γ are *mutual information-based feature selection* [3], *maximum-relevance minimum-redundancy criterion* [86], *joint mutual information* [128], *mutual information uniformly distributed* [67], *conditional info-max* [71], *conditional mutual information maximization* [32] and informative fragments [117]. Moreover, it becomes easier to compare the sensitivity of such families of methods with respect to redundancy and noise.

2.6.2 Feature extraction

We define *feature extraction* [45] as the generation of new features by combining all or some of the existing ones. A common way to extract features is based on expert knowledge, but we will not consider it as it is subjective to the problem and is not fully data-driven.

The most relevant feature extraction approaches are based on the already mentioned projections on latent subspaces. The core methods are PCA and its many variants. In the original PCA the number of extracted features is the same as in the original data, since each principal component is just a linear combination of all the original features and there are as many linear combinations as the original number of features, thus PCA is a linear method. Assuming mean centered data, from a linear algebra viewpoint it consists just on a rotation of the coordinate axes.

The gain when applying PCA is that the new features (principal components) are ordered from higher to lower amount of captured variance in the set of features in the original data (ignoring the target, i.e. unsupervised). The cumulative variance captured by nested subsets of PCs can be easily computed, allowing to set a cut threshold in the number of PCs, leading to a reduction in the number of fea-

tures (data compression) in such a way that the variance that is left out is small and controlled, possible colinearities between features are overcome, and, theoretically, noise is filtered.

Even when it contradicts intuition, compression is not always a goal in preprocessing when applying PCA. A situation in which it is not worthy, even counterproductive, to compress is noise-filtered data to be used afterwards by an algorithm that internally includes an embedded feature selection, such as random forests [10]. It is not a rare situation in predictive maintenance applications because noise filtering through transformations is well-established.

Many variants are motivated by a non-linear nature of the data. For instance, if there are certain known/intuited non-linear relations between samples somehow having similar consequences as colinearity, we could model them by means of a specific kernel function and apply KernelPCA [97]. In this way, by using the kernel trick, we transform our feature space into a space where those relations look linear, applying there PCA.

As an alternative to the philosophy behind PCA-based approaches, we can consider neighborhood embedding approaches that try to preserve local neighborhood structures in the data on lower dimensionality spaces. A well-known algorithm for neighborhood embedding is *Stochastic Neighborhood Embedding* [53] (SNE) in which a Gaussian probability distribution describes the potential neighborhood of each original sample in the high-dimensional space. A variation of SNE, with a simpler optimization process and comparable performance, is *t-distributed SNE* [75] (t-SNE). Despite it was originally developed for visualization purposes, it is perfectly applicable for data compression.

Both SNE and tSNE are non-linear algorithms. A linear method also meant for neighborhood preservation is *Locality Preserving Projections* [52] (LPP). In the same paper, the authors propose a non-linear extension, named Kernel LPP, just by applying the kernel trick before LPP.

In highly dynamical systems, as is the case here, it is an adequate strategy to perform several local linear models covering the zones of influence of the data as a way to obtain the behavior of a non-linear global model by aggregation/ensemble [130] of the local linear ones. A use of this technique, in which the aggregation of the local linear models is achieved by means of a fuzzy inference system [74] can be found in [13]. The authors aplied the strategy for regression purposes, but it could be adapted for feature extraction using local information. In case of favorable properties in the aggregation algorithm, this strategy is also suitable for on-line monitoring[14, 16].

2.6.3 Feature discretization

Some of the most famous algorithms employed in machine learning in general, thus also in monitoring in predictive maintenance, are meant for categorical values (e.g. decision trees). Moreover, sometimes they can only handle such type of data. Besides, the type of data in predictive maintenance applications consist of numerical continuous features with an order relationship, e.g. sensor data. Therefore, it makes sense to think of ways to transform such features into categorical ones, so that those algorithms could be used. This procedure is called *discretization*, and it is performed feature by feature independently.

Assuming we have a feature X_i whose values are numerical values with an order relationship. If we denote the minimum value by m, and the maximum value by M, then a discretization process consists in the definition of K intervals

$$I_1 = [a_0, a_1), I_2 = [a_1, a_2), \ldots, I_{K-1} = [a_{K-2}, a_{K-1}), I_K = [a_{K-1}, a_K]$$

where $a_0 = m$ and $a_K = M$. Notice that the cutpoints for the intervals define the partition of the range unambiguously.

A naïve approach would be to prefix K and split the range $[m, M]$ into K equal-length intervals. There are several drawbacks with this method. First of all, which is the right value for K? If the data is sparse or some extreme values (outliers or not) are present, then the range is huge. In such situation it could happen that certain intervals are empty and some crowded. Therefore, unless our data is uniformly spread and we have a proper way to choose K, it is not a good option. Nevertheless, there are plenty of estimators for the width (W) of the bins, thus for K. One robust option is the *Freedman-Diaconis rule* [33], given by

$$W = 2 \cdot \frac{IQR(X)}{\sqrt[3]{N}} \tag{18}$$

where X is the feature under consideration and N the number of samples.

Hence, it is preferable to have a clever way to proceed that, if possible, does not force us to prefix K. The widest used method is a supervised top-down algorithm called *minimum description length principle (MDLP)* [30]. By top-down we mean that it begins with an empty partition and the cutpoints are added on the fly, thus no need to prefix K. It decides whether a new cutpoint is needed and where to locate it by means of information theory, concretly the mutual information with the target [103].

There are many other ways to define discretizations. An exhaustive survey including several taxonomies according to the properties of the methods and the data is available in [36]. The authors present 87 methods, tested on many datasets with different

properties, so by comparison of types of data we can have a guess of which methods could fit better to our data.

2.7 Imbalanced data treatment

In such cases when the data show a lack of balance between the classes of the samples, it is usually the case that the class we are more interested in is the minority class, e.g. faulty and fault-free samples. Despite we have commented in Section 2.1 on two ways to deal with imbalanced data, named as sampling- and weighting-based; the latter is more related to the modeling phase instead of the preprocessing phase because the weights are actually introduced in the model creation or the model validation steps, depending on the characteristics of the algorithm that is being employed.

It is also of uppermost importance the metrics employed in the validation. For instance, accuracy is not a valid choice because if the imbalance is 99%-1%, then predicting the majority class always leads to a 99% accuracy. Thinking on the example of faulty and fault-free samples, we would predict that faults never happen, being almost always right. But it is obvious that not all errors in our prediction have the same cost. In order to mitigate this without a need to assign a cost per error it is common to use ROC curves [31]. Nevertheless, this is out of the scope of this chapter, as it does not correspond to preprocessing. Therefore, we focus on imbalance treatment approaches based on sampling techniques.

There are two obvious ways of compensating the imbalance, which are adding samples from the minority class (*oversampling*), and removing samples from the majority class (*undersampling*) [108, 121, 59, 87]. Because both have pros and cons [19, 27, 79], it is also common to opt for a hybrid approach (*mixed sampling*) [66, 2] combining them.

Recently, the importance of ensemble methods has been shown in many applications including imbalance treatment [20]. Both ensembles of repetitions of stochastic techniques or ensembles of diverse deterministic techniques usually overcome the application of single techniques, ensuring robustness by reducing the variance.

2.7.1 Oversampling

If we think on how to perform oversampling, the first intuitive approach is *random oversampling* (with or without replacement). In [58] the authors consider two random oversampling possibilities: a pure random one and a focused one in which only samples close to the boundary between classes are considered as selectable; both used until parity in the classes is reached. As not all the samples from one class

influence the monitoring algorithm in the same way, the samples we replicate could be so influential that we suffer from an overfitting effect. More sophisticated approaches opt for creating new samples by interpolation of some of the existing ones.

There are two main methods, existing several variants for each of them. Those relevant methods are *synthetic minority oversampling technique (SMOTE)* [18], and *adaptive synthetic sampling method (ADASYN)* [51]. In both methods the algorithm to generate new samples is the same. A sample x_i from the minority class is considered. Then the K-nearest neighbors from the minority class are located. One of them x_j is randomly chosen, and the new synthetic sample is a convex combination of them

$$x_{new} = \lambda x_i + (1-\lambda)x_j$$

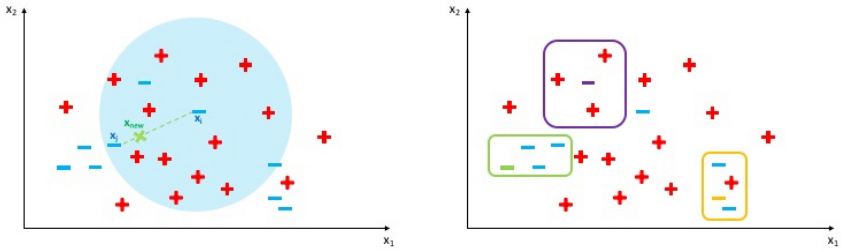
where $\lambda \in [0,1]$ is randomly selected. Graphically, the convex combination of two points is a point located in the segment that joins them. Figure 3 shows the generation of a new minority class sample x_{new} (marked as a green cross). The difference between SMOTE and ADASYN is only in the way the neighbor points are taken. The latter uses a prefixed K, and the former chooses it depending on the density of the minority class inside a neighborhood obtained by K'-nearest neighbors.

The influence of extreme values (or outliers, if not detected) is really high in both SMOTE and ADASYN, being higher in ADASYN. Then SMOTE is usually employed in some of its variants. The most famous ones are *borderline-1 SMOTE*, *borderline-2 SMOTE* and *SVM SMOTE*. In borderline versions also an auxiliary K' neighborhood is used, where the samples x_i from the minority class are labeled as *noisy* (all nearest neighbors are not from the minority class), *in danger* (at least half of the neighbors are from minority class), or *safe* (all are from the same class as x_i). Then the only samples chosen as initial samples are *in danger* samples.

The difference between borderline-1 and borderline-2 happens when selecting x_j. Borderline-2 allows to select a sample from any class, not necessary majority class (as borderline-1 does). In SVM SMOTE the support vectors are used to generate the new sample x_{new}.

2.7.2 Undersampling

The major risk when ignoring majority class samples is to potentially ignore really relevant informative samples, leading to a degradation of the general quality of the model. As in oversampling, there are methods that select (sample selection) prototypes in the majority class (most of the approaches) and methods that generate (sample extraction) a smaller set of prototypes from the original bigger set of samples. The only relevant approach in *prototype generation methods* is called *cluster centroids undersampling*, which is based on clustering using representatives (CURE) [42], a famous clustering algorithm in which relevant points of the identi-



(a) Synthetic sample generation. (b) Noisy, borderline, and safe points.

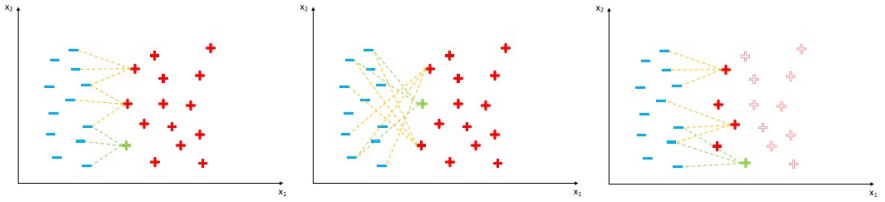
Fig. 3 Examples of (a) the generation of a new minority class sample x_{new} from an existing sample x_i and one of its 3-NN x_j (the 3-NN neighborhood of x_i appears as a pale blue circle), and (b) the determination of noisy (purple), borderline (orange) and safe (green) minority class samples in borderline extensions of SMOTE (the 3-NN are inside colored rounded squares). Plus and minus symbols represent majority and minority class samples respectively.

fied clusters (e.g. the centers) substitute the points inside those clusters, reducing the amount of points but keeping the underlying cluster structure. When it comes to *prototype selection methods*, we can identify two subgroup of methods depending on the possibility by the user of controlling the number of samples after undersampling (*controlled undersampling techniques*) or not (*cleaning undersampling techniques*).

The simplest *controlled undersampling* technique is *random undersampling*, which is the riskiest one as all samples are equiprobably deleted ignoring their potential informativeness/relevance. The most representative approach is called *NearMiss* [131], which includes some heuristic rules in order to select the samples. The authors presented three NearMiss versions, differing in the way the heuristics are defined. *NearMiss-1* selects the majority class samples with minimum average distance to the N closest minority class samples. *NearMiss-2* selects the majority class samples with minimum average distance to the N farthest minority class samples. Finally, *NearMiss-3* has two steps: first, the M nearest neighbors for each minority class sample are kept, then the majority class samples with maximum average distance to the N closest minority class samples are selected.

Also in [131], the authors define another approach, named *MostDistant*, in which the selected majority class samples are those presenting largest average distances to the N closest minority class samples. In the original paper the authors select $N = 3$. Figure 4 shows examples of the three versions of NearMiss in a two-dimensional space with $N = 3$ and $M = 5$.

The family of *cleaning undersampling* techniques is bigger. The name comes from the fact that the part of the dataset corresponding to the majority class is cleaned by deleting certain samples considered as dispensable according to certain heuristic algorithm. We describe them in no particular order.



(a) NearMiss-1 (b) NearMiss-2 (c) NearMiss-3

Fig. 4 Examples of the selections performed by all three NearMiss versions. Plus and minus symbols represent majority and minority class samples respectively. Distances to the 3-NN of some majority class samples are depicted using colored dashed lines. In green we can see the distances corresponding to the selected majority class sample, as well as the sample itself in each version. In (c), the samples out of the 5-NN neighborhood are represented.

A popular method is based on the so-called *Tomek's links* [112]. We say that two samples from different classes form a Tomek's link if they are nearest neighbors to each other. Mathematically,

$$d(x,z) \geq d(x,y) \text{ and } d(y,z) \geq d(x,y), \forall z \tag{19}$$

The undersampling procedure associated to them has two variants. We can remove (i) only the sample in the Tomek's link corresponding to the majority class, or (ii) both samples. It is clear that such pairs of samples are some sort of contradiction. The safest choice would be to remove both, but this could be sometime not an option as it would decrease the size of the minority class. An example of a Tomek's link can be seen in Figure 3(b), formed by the purple minority class sample and its nearest neighbor.

Inspired by Wilson's studies on the nearest neighbors rules [123], *edited nearest neighbors* edits the dataset by removing those samples which do not *agree enough* with their neighborhood. Different agreement criteria provide different versions. Given one sample, the most restrictive version demands all the samples in the neighborhood to be from the same class of the sample under study. A more relaxed version, demands only a majority of samples from the same class. There is also the possibility to run the edition procedure several times iteratively (with the same or different K), so that more samples are removed.

We can define *instance hardness* [106] as the level of difficulty to predict the class of a sample due to the sample characteristics. The usual way to calculate it is by means of an algorithm that assign to each sample, using cross-validation, a probability of being well classified, thus the lower the probability the harder the instance. The instance hardness undersampling technique consists in establishing a threshold for the probability of the majority class samples, removing those that are below the threshold.

Last but not least, we have the family of *condensed nearest neighbors*, based on the homonymous rule [47]. The undersampling methods that are inspired on it condense the space by removing samples that are far from the decision boundaries. The original method is based on an iterative process with the following steps

1. Construct a *condensed set C* containing the minority class samples.
2. Add one majority class sample to C, and create a *potential set P* with the rest.
3. Classify each sample in P using 1-NN. If misclassified, move it to C. Otherwise, do nothing.
4. Reiterate until no samples can be added to C.

As this original approach is very sensitive to noisy samples, keeping them in C, some variants were proposed. The variant named *one-sided selection* [66] removes noise by applying Tomek's links first, and then the steps 1. to 3. of the original approach, thus no iteration over P.

In [68], the authors propose *neighborhood cleaning rule*, that proceeds as follows

1. Get one sample x_i and classify it using 3-NN.
2. If x_i is misclassified, go to next step. If classified go to the first step.
3. If x_i is a majority class sample, then remove x_i. If x_i is a minority class sample, then remove the 3-NN corresponding to the majority class. Go to the first step.

This approach is computationally expensive, and could suffer in case of very large heavily imbalanced datasets. Even when its philosophy is based on cleaning, the result is usually a condensed subset of the original one.

All these condensed family techniques depend on some randomness, when taking samples to begin. Moreover, the order of the samples is relevant for the final undersampled set. Therefore, we cannot expect the same result when repeating them over the same dataset. It is recommended to perform the methods several times and ensemble the results by certain aggregation procedure.

2.7.3 Mixed sampling

The naïve approach, consisting on combining both random oversampling and random undersampling was proposed in [72]. The authors used lift analysis instead of accuracy as performance score measurement in their experiments, without obtaining relevant improvements.

A deep study on the mixture of oversampling and undersampling techniques can be found in [2]. The authors point out the good results of mixing SMOTE both with Tomek's links and edited nearest neighbors.

2.8 Models

Our aim is not checking which modeling technique behaves better, but comparing different preprocessing schemata by mean of the posterior performance in a regression or classification task. Therefore, we present only a few techniques just to check whether using different algorithms is also relevant in the selection of the right preprocessing scheme apart from the data.

Here we briefly describe some state of the art algorithms suitable for confronting predictive maintenance problems. We distinguish to types of algorithms for two classical problems. Classification algorithms for fault detection problems, and regression algorithms for remaining useful life prediction problems.

2.8.1 Classification

A regular fault detection problem is a binary *classification* problem in which the aim is to predict whether a concrete system state (sample) corresponds to a faulty or to a fault-free situation. The simplest but still widely used classification methods are naïve Bayes and K-nearest neighbors. *Naïve Bayes* (NB) algorithm [34] is a probabilistic method based on the application of Bayes theorem under strong feature independence assumptions. K-nearest neighbors algorithm [105], as all methods based on distance calculations, can suffer from huge distances of some of the neighbors due to the sparseness enforced by a habitual high dimensionality. The attempts to mitigate such problem is the motivation behind *distance weighted K-nearest neighbor* algorithm [28], that is the variation of K-NN we will consider, consisting in regulating the importance of the votes of the neighbors by means of weights that depend on the distance, so that the closer the more important. Since it is the only variant we will consider, we denote it by *K-NN*.

Support vector machines (SVM) [115, 98] is a well-known nonlinear classification method, based on separating the classes employing hyper-planes is such way that the separation is maximized. This separation is not performed in the original input space but in a kernel-transformed space, i.e. the kernel trick [55]. The samples that are closest to the decision boundary, thus defining the hyper-planes, are called *support vectors*. In [62] the authors compare several classifiers for fault detection, including distance-weighted K-nearest neighbors and support vector machines among others.

The *random forests* (RF) algorithm [10] is a stochastic ensemble method that performs a bagging strategy (a combination of bootstrapping and aggregation [9]) of weak learners, concretely decision trees. The procedure is simple. Given a prefixed a number of trees, for each tree a subset of the original features is randomly selected (*weakness*). Then the tree is trained using those features and a set of samples obtained by random selection with replacement (*bootstrapping*). The decision is ob-

tained by combining all individual tree decisions (*aggregation*). The magic behind RF is that the bias of the full ensemble is equivalent to the bias of each single tree, whereas the variance is much smaller. This robustness, together with its low computational cost and high parallelization and distribution capabilities makes RF an algorithm to be taken into consideration in predictive maintenance [43, 127, 12].

2.8.2 Regression

Despite the original purpose of RF and SVM is classification, there are versions of both of them for regression purposes. In the case of random forests, it is quite straightforward to substitute decision trees by regression trees, and the voting aggregation by an average prediction [10]. The insights in the case of support vector regression (SVR) are a bit more complex and too long to be commented here [107]. Some applications to RUL prediction can be seen in [125, 90, 4, 73].

Basic linear regression approaches, such as multiple linear regression, suffer from the arising of singularities because of the effect of colinearities between features when calculating the inverse of $X^T X$, required by the least squares solution, being X the input data matrix. In such situations, shrinkage (regularization) methods avoid singularity by perturbing the matrix before it is inverted. The two main approaches in the family of shrinkage methods are *Lasso* [111] and *ridge regression* [49], obtained by introducing ℓ_1 and ℓ_2 penalties respectively. The *elastic net* [133] includes a penalty based on a combination of both ℓ_1 and ℓ_2 penalties, looking for some elasticity in the regularization, being Lasso and ridge regression particular cases of the elastic net.

Generalized linear models [49] is a generalization of ordinary linear regression that provides flexibility in the sense that the distribution of the errors is not necessarily supposed to be normal, as happens in ordinary linear regression. The combination of the elastic net with generalized linear models (GLMnet) is a regression algorithm based on generalized least squares that uses cyclical coordinate descent [50] in a path-wise fashion [48] in order to select the optimum elasticity in the regularization via the elastic net. The elasticity provided by the possibility of controlling how close we are to Lasso or ridge regression by means of a single parameter allows an efficient exploitation of the regularization benefits.

Up to our knowledge, this approach has not been used in predictive maintenance yet. Nevertheless it has been considered here because of its outstanding results in monitoring dynamic chemical systems in process analytic technology (PAT) [17, 15], that behave quite similarly to regression problems in predictive maintenance with dynamic systems.

Deep Learning (DL) is the way to call the use of a complex artificial neural network. A neuron is a single computation unit that receives an input value (from a

data source or another neuron), performs a simple operation consisting on applying certain simple function (*activation function*) over the product of the input by a numerical parameter (*weight*), and outputs the result (towards an output interface or a neuron).

Different types of neurons connected in different ways lead to different network architectures. These neural networks are designed by means of layers of neurons conceived for specific subtasks.

For stream-like data, such as the data usually involved in monitoring tasks in predictive maintenance, the most used networks are *recurrent networks* (RNN), in which the neurons are also connected to themselves. This provides the network with some *memory* in the form of persistence of the information. In general, they suffer when the ideal persistence time grows.

There is a family of RNNs meant for handling long term information dependencies called *long short term memory networks* (LSTM) [54] that contain an internal mechanism (*cell state*) to filter/retain part of the information as long as necessary. There are several ways of handling the remembering/forgetting part of the learning process, leading to different variants of LSTMs. The most relevant ones are, amongst others, *Vanilla LSTM* [40], *Gated Recurrent Unit* (GRU) [21], *Depth Gated LSTM* (DG) [129] or *Grid LSTM* [64]. In principle, LSTM networks are the most adequate network architectures in predictive maintenance.

Even when the natural output of the network is a number, they could be adapted for classification purposes by linking the classes to certain numerical output ranges.

3 Experimentation

The philosophy derived from non-free-lunch theorem [124], which states that the average performance of all algorithms over all possible problems is asymptotically the same, is that there is not a single universal algorithm that is the best. Therefore, there is always margin for improvement and every particular problem (correspondingly dataset) is better suit for a different method. This applies also to the preprocessing schemata, in the sense that there is not an universal preprocessing schema that is always the best, being the goodness problem/data dependent.

Consequently, providing successful stories for concrete scenarios is perhaps not the best option. It would be more relevant to provide the reader with direct or literature referenced details of the available choices in the market, as well as hints about possible decisions depending on the characteristics of the problems or the data. For such reason we will just employ the already presented classification and regression techniques on some of public available real-world datasets from competitions in the

Annual Conference of the Prognostics and Health Management (PHM) Society. We will use them (both the original data and some modified subsets, e.g. for missing values treatment or outlier detection) to compare several preprocessing schemata on different algorithms. Furthermore, we will provide some clues about which preprocessing methods might be more reasonable depending on the particularities of data and problems based on successful applications.

It is obvious that the combination of all possible methods in all steps in different orders would end up in thousands of preprocessing schemata. Moreover, if a schema consisting on seven steps works very well, we would not be able to decide which of them contributed more to that behavior. Therefore, just some schemata involving only a few steps will be tested, and compared also with, we should not forget that it is always a possibility, not preprocessing at all.

3.1 Datasets

In order to have a classification and a regression problem, we have considered the data corresponding to the *PHM Data Challenge 2014* and the *PHM Data Challenge 2016*. The former is transformed into a fault detection problem (classification), and the latter is a RUL estimation problem (regression) in which the average removal rate of material in a polishing process. The lack of exact environmental information about the origin of the datasets impedes us to infer cause-effect reasons for the results. Hence, we focus on the goodness of the application of the preprocessing schemata instead of the underlying reasons.

3.1.1 PHM challenge 2014

The information about the domain and the data of the *PHM challenge 2014* is not provided due to proprietary concerns. We know that it consists of 6 datasets, half for training and half for testing with information about (i) *part consumption* (i.e. the replacement of some parts), (ii) *usage* (similar to the lines of an odometer), and (iii) *failures* (time of failure). The target information for the test files in unknown, so we focus only on training data. By crossing the failure information with the rest we could build by merging a dataset in which the target is binary: faulty or non-faulty, thus it consists on a binary classification problem. For further information on the data, check the call for participation in [37].

In this dataset there is almost no information about the nature of the features. The original aim in the challenge was to predict the health level of the components in a certain time by classifying them into low risk and high risk of failure, equivalent to fault save and faulty in a short future. The variables are numerical and discrete. Nevertheless, the amount of different values is so big that we can employ any method

suitable for continuous variables.

The data is heavily imbalanced, belonging to the high risk class (faulty) only 4% of the samples. The modeling algorithms does not take into account the level of imbalance during training. Only some of them, based on iterative optimization processes, are capable of weighting the errors according to the class densities so that they favor avoiding mistakes in the minority class. The main problem of such approaches is the price to pay in the prediction of the majority class. Therefore it is recommended to treat imbalanced in advance as part of the preprocessing.

3.1.2 PHM challenge 2016

The system under investigation is a wafer Chemical-Mechanical Planarization (CMP) tool that removes material from the surface of the wafer through a polishing process. Figure 5 depicts the CMP process components and operation. The CMP tool is composed of the following components: (i) a rotating table used to hold a polishing pad, (ii) a replaceable polishing pad which is attached to the table, (iii) a translating and rotating wafer carrier used to hold the wafer, (iv) a slurry dispenser, and(v) a translating and rotating dresser used to condition a polishing pad.

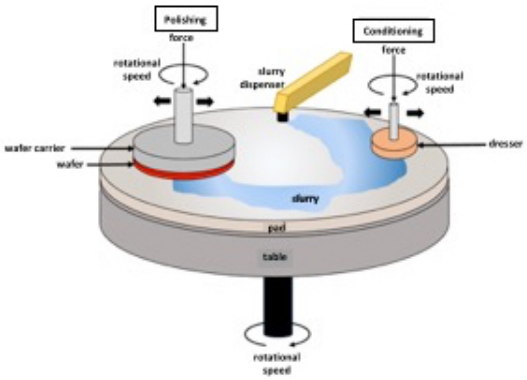


Fig. 5 Chemical Mechanical Planarization (Polishing) of wafer.

This process removes material from wafer surface.

This image is property of the Prognostics and Health Management Society and was taken from the online information about the challenge.

During the polishing process, the polishing pads ability to remove material is diminished. Over time, the polishing pad has to be replaced with a new pad. Similarly, the dressers capability to roughen the polishing pads is also reduced after successive conditioning operations and after a while the dresser must be replaced. The objective is to predict polishing removal rate of material from a wafer, thus it is a regression problem. For further details, check the call for participation in [88].

A deeper look at the data allows as to infer some characteristics of the data. All

variables are numeric (float), with different ranges and dynamics. Some fluctuate up and down approximately in a cyclic way while some others show a continuous increase or decrease that is apparently linear. These differences force us to be careful when selecting the way to apply the preprocessing techniques.

For instance, if we apply a technique that involves mixing the features, such as PCA, then standardization is recommended. On the contrary, in approaches acting on the features individually, such as discretization, it could be counterproductive. Due to the size of the data we are limited to visualization techniques based on certain information summary/compression such as tSNE plots, and scores and loadings plots in PCA. Nevertheless there is not an obvious relationship between the visualizations and the adequate preprocessing techniques.

3.2 Experimental schema

Since our algorithms are favored by centered data or translation invariant, then we mean centered all the data in advance. All the experiments were made using 10-fold cross validation because it is known to be a good approximation of the expected prediction error on separate future test samples. We evaluate the performance in classification by means of the area under the ROC curve [6], and in regression with the root mean square error.

Since our intention is not to beat the winners of the competitions, but check whether preprocessing is beneficial or not (and how much), then our comparisons are against not preprocessing. The reason for including several modeling algorithms is not to determine which one is better, but to try to check if that diversity of models is relevant or not for the benefit of preprocessing.

In case we suspect that the best preprocessing strategy is independent of the posterior modeling technique, then we could try the simplest ones in order to guess the right preprocessing scheme. For statistical significance of the differences, we have employed the *Mann-Whitney-Wilcoxon* test [80].

In this study both the outliers and missing values have been artificially introduced, thus we have the chance to check the performance of the methods, as we know the truth. This is not the case in real-world applications. With respect to the rest of preprocessing steps, we have performed the test with the full dataset. The realist approach in an application would be to extract a representative subset of the data in order to perform some preliminary tests and determine a full preprocessing strategy.

When it comes to the study of the approaches for missing values, we have modified the PHM2016 dataset by randomly erasing 1% of the values in 10% variables. Taking into account that there are 21 variables and 346015 samples, we have in-

troduced in 2 variables 3460 holes per variable. The approaches employed were imputation with the mean value, imputation by averaging using 5-NN, as well as removing the samples (deletion strategy).

For outlier detection, also using PHM2016, we have modified 1% of the total amount of single numerical values by distancing them from the mean of the feature they correspond to. The amount of variation is proportional to their distance to the mean, with factors corresponding to 20%, 50%, 100% and 200%, meaning 865 variations per level. Each of them has been applied to the one fourth of the modified values, i.e. 0.25% of the total amount of values. In this way we can evaluate the sensitivity to the amount of variation. The approaches employed were the Mahalanobis distance and the Q and T^2 approximations in their off-line versions.

As some potential detected outliers could be out of the list of the artificial modifications (false positives), it makes sense also to check the posterior performance, after cleaning, in modeling. For this comparison, we have also included the original modified data, i.e. without looking for oultiers.

When it comes to feature engineering, we have designed experiments separately for feature selection (on PHM2016), feature extraction (on both datasets), and discretization (on PHM2016). The algorithms we have employed are:

- **Feature selection**. We have chosen two filter methods (correlation-based feature selection, and conditional info-max), and a wrapper approach (using K-NN).
- **Feature extraction**. We have selected PCA, PLS, Kernel-PCA, t-SNE and Kernel-LPP, so we have two linear and three non-linear methods. Notice that most of the features in PHM2014 are numerical discrete variables containing natural numbers. Nevertheless, the amount of different values is so big that we can consider them as continuous numerical variables, suitable for feature extraction by PCA or PLS. The adequate number of *PC*s and *LV*s have been selected by grid search.
- **Discretization**. We have opted for two approaches, in order to consider one that prefix the number of bins (equal-width intervals using Freedman-Diaconis rule), and another one that does not prefix it (MDLP).

For *imbalanced data treatment* we need a classification problem, thus we use our PHM2014 version, which has a minority class (faulty) represented approximately by a 4% of the data samples. We have not applied all the methods in Section 2.7, but some of the most popular ones. Classified according to the provided taxonomy, they are

- **Oversampling**. Random oversampling, SMOTE borderline-2, and ADASYN.
- **Undersampling**. Random undersampling, cluster centroids, NearMiss-2, and Tomek's links.
- **Mixed sampling**. Random oversampling and undersampling combination, and SMOTE with Tomek's links.

3.3 Results

The results are presented by means of tables, which formats depend on the experiments. As general facts,

- we consider not preprocessing as baseline, and we present the percentage of improvement (positive number) or deterioration (negative number). An exception occurs in the case of missing values, because the usual baseline does not exist. In that case we consider the deletion strategy as the baseline approach. If the shown variation from the baseline is significant, according to the Mann-Whitney-Wilcoxon test, it will be indicated with a ‡ mark. The best results are highlighted in **bold** font. There is also another exception when studying the detection of outliers. In that situation there is not any baseline because there is not any modeling step, but just checking the performance in the detection of the outliers for different perturbation levels. In this case we just show the detection rates per method and per level, and the † mark means significantly better than the other method;
- in all nearest neighbor related approaches in which we have the chance of choosing K, our choice will be $K = 5$;
- the kernel function used in both SVM and Kernel-PCA is radial basis function (RBF);
- the network architecture used for DL is GRU because it has a simple effective joined input/forgetting mechanism by using the so-called *update gates*, proved to behave similarly to much more complex arquitectures [41];
- unless explicitly indicated otherwise, the learning parameters of the algorithms are set by default as in the literature. For GRU, the default arguments in Keras [22] have been used.

Table 1 shows the results for missing values. Notice that in this situation all columns are independent because we are comparing, for each modeling technique, the performance of imputation versus deletion for that concrete technique. For instance, the values $+1.35$ and $+2.57$ corresponding to RF algorithm mean that imputation is preferred (both are positive values) and the performance when using K-NN method is almost doubly beneficial than Mean.

We can see that it is slightly beneficial to use imputation, being a bit better the imputation by means of K-NN. Nevertheless, none of the imputations are statistically significantly better than deletion, for any algorithm except for RF and GRU. This, together with the fact that K-NN requires huge computational and memory resources, shows doubts about its suitability.

The reason for using several models is to check whether the model to be applied after preprocessing has an impact in the right preprocessing scheme. Hopefully we can see that the results are similar for all modeling algorithms, thus it seems that the data is more relevant than the algorithm. Nevertheless, we should notice that there are big differences in performance between deletion, Mean and K-NN for the various modeling techniques even when the general trend remains stable.

Table 1 Missing values.

Method [a]	RF	SVR	GLMnet	GRU
Mean	+1.35	+0.15	+1.20	+1.57 ‡
K-NN	+2.57 ‡	+0.33	+1.28	+2.04 ‡

[a] Deletion strategy is considered as the baseline.

Table 2 shows the results for the detection of outliers for different deviations. The percentage of outliers is constantly 1%, but the amount of deviation from the original values, artificially introduced, varies from low intensity (20%) to high intensity (200%). The higher the intensity the simpler the detection because the values are much more different from the real ones. In the case of the Q and T^2 approximations method, the number of principal components has been determined by establishing a threshold of the total amount of variance captured, set in 90%.

In general, the approximation approach behaves better than Mahalanobis, being that difference higher in the intermediate levels. For the biggest distortions (easier to detect) both methods perform very well.

Table 2 Outlier detection accuracy.

Method	20%	50%	100%	200%
Mahalanobis	1.04	13.87	53.29	**92.37**
Q and T^2 [a]	**1.62**	**26.82** †	**69.71**	91.91

[a] The number of *PC*s is 4.

Table 3 shows the results for outlier detection effect in modeling. Looking at RF column we can see that the advantage is much lower than for the other two algorithms. A possible reason is the fact that RF uses for each tree a reduced dataset, both in the features and in the samples part. Theoretically the expected percentage of the samples from the original set considered for training each tree is indeed 63.2%, thus errors in the detection could be somehow partially neglected.

Besides, we could suspect the difference between SVM (non-linear) and GLMnet (linear) to be due to the fact that the transformation used for generating the outliers is a linear mapping. Nevertheless, the suspicion is not right because GRU is also non-linear and behaves almost the same as GLMnet. In the end, GLMnet and GRU have suffered less than SVM from the not detected outliers or the false positives. The latter are very few, almost zero compared to the true outliers.

Table 4 shows the results for feature selection. The most plausible reason for the total lack of advantage in this feature selection process is that the variables are quite

Table 3 Outlier detection effect on modeling.

Method	RF	SVR	GLMnet	GRU
Mahalanobis	+0.66	+2.09	+4.10 ‡	+3.96
Q and T^2 [a]	**+0.92**	**+2.33**	**+5.01** ‡	**+5.14** ‡

[a] The number of *PC*s is 4.

independent, containing a similar amount of complementary information. Therefore, selecting features in any way enforces certain information loss. The effect is magnified in RF, as it has an internal tree-wise feature selection step.

Table 4 Feature selection.

Method	RF	SVR	GLMnet	GRU
CFS	-0.77	+0.03	+0.38	+0.20
Conditional info-max	-0.07	**+0.05**	+0.25	+0.16
K-NN [a]	**+0.02**	+0.03	**+0.41**	**+0.22**

[a] Only wrapper method. The rest are filters.

Tables 5 and 6 show the results for feature extraction in the classification and regression tasks respectively. We can say that (i) the data seem to be quite non-linear, as the non-linear methods are the best in all algorithms except SVM in classification (probably due to the fact that in that case we are applying twice an equivalent kernel trick), (ii) PLS (supervised) behaves better than PCA (unsupervised) because of the possibility of using the target information, and (iii) it makes sense to use these feature extraction methods, even when the improvement is not statistically significantly better (with one single exception), because they are not much computationally expensive.

Table 5 Feature extraction in fault detection (classification).

Method	NB	K-NN	SVM	RF
PCA [a]	+1.03	-1.11	+1.48	+1.27
PLS [b]	+1.24	-0.22	**+2.17** ‡	+1.33
Kernel-PCA [a]	**+1.36**	+1.04	+2.04 ‡	+2.16 ‡
t-SNE	+1.25	**+1.53** ‡	+2.11 ‡	**+2.35** ‡
Kernel-LPP	+1.21	+1.15	+2.16 ‡	+2.08 ‡

[a] The number of *PC*s is 3.
[b] The number of *LV*s is 2.

Table 7 shows the results for discretization. In this case the comparison between equal-width and MDLP is not totally fair, as the former is unsupervised and the latter supervised. Also, in general, methods that do not need to prefix the number of

Table 6 Feature extraction in RUL estimation (regression).

Method	RF	SVR	GLMnet	GRU
PCA [a]	+2.20 ‡	+1.26	+1. 52	+2.15 ‡
PLS [a]	+2.31 ‡	+1.53	+1.59	+3.02 ‡
Kernel-PCA [a]	+3.33 ‡	+1.54	+2.60 ‡	+3.48 ‡
t-SNE	**+4.22** ‡	**+1.78** ‡	**+2.85** ‡	+4.39 ‡
Kernel-LPP	+3.97 ‡	+1.60	+2.71 ‡	**+4.81** ‡

[a] The number of *PC*s and *LV*s is 4.

bins achieve results at least as good as the restrictive ones. In this case, according to the significance tests, it is clearly not an exception, especially in RF, GLMnet and GRU. Equal-Width only makes sense if the density of the features is homogeneous in their ranges, which is rare and not happening here.

In general, RF use to behave better when discretizing. Besides, some algorithms involving complex/computationally expensive optimization processes (like SVR and GRU) could suffer from numerical instabilities that are less likely with discrete features. Nevertheless, discretization is not necessary beneficial always. Also notice that the process affects the features independently, which makes possible to discretize only a subset of the continuous features.

Table 7 Discretization.

Method	RF	SVR	GLMnet	GRU
Equal-Width [a]	-1.61 ‡	-0.32	-0.95	-2.33 ‡
MDLP	**+2.66** ‡	**+0.98**	**+2.76** ‡	**+4.19** ‡

[a] Using Freedman-Diaconis rule.

Table 8 shows the results for imbalance data treatment. First, we can compare this results with the ones in Table 5, and point out that imbalanced data treatment schema seems to be a better choice than feature extraction schema, hence, as we have mentioned before that it was worthy to use feature extraction, it is even worthier to use imbalanced data treatment.

Looking only at these imbalanced methods, it is clear that all three random approaches are a bad choice, independently of the algorithm employed. Maybe the flexibility provided by its non-linear nature makes SVM be the least bad. It seems logical that SMOTE+Tomek is the best when SMOTE was the best among the oversampling methods and Tomek's links among the undersampling methods. In this case it has occurred, but it is not always necessary the case.

Table 8 Imbalanced data treatment.

Method	NB	K-NN	SVM	RF
RandOver	-1.83 ‡	-2.01 ‡	-0.92	-1.24
SMOTE [a]	+1.39	+1.06	+2.43 ‡	+1.40
ADASYN	+0.23	-0.82	+1.49	+0.48
RandUnder	-1.71	-1.69	-1.03	-1.43
ClustCentr	-0.02	-0.85	+1.04	+0.62
NearMiss [a]	+1.22	+1.03	+1.15	+1.55
Tomek	+1.42	+0.99	+1.24	+2.51 ‡
RandOverUnder	-1.75	-2.25 ‡	-1.20	-1.37
SMOTE+Tomek	**+1.81**	**+1.13**	**+4.61** ‡	**+3.49** ‡

[a] The version is SMOTE borderline-2.
[b] The version is NearMiss-2.

4 Conclusions

We have presented in detail methods covering all steps involved in preprocessing in predictive maintenance, both for offline and online learning scenarios when the latter was feasible, as well as provided the reader with exhaustive bibliographic references.

We have performed several experiments on public available real-world data from the PHM Data Challenges 2014 and 2016, so that we could empirically test some of the presented approaches.

We have seen that the data seem to have higher relevance than the posterior modeling technique in order to determine the preprocessing schema, both for regression and classification problems.

As possible extensions, some online tests could be performed, in order to check the online versions of the methods. Besides, more preprocessing strategies, modeling algorithms and datasets could be considered in order to extend the study and check with higher certainty whether the modeling algorithm is much less relevant than the data for the adequate preprocessing scheme.

Acknowledgements This research is supported by the Basque Government through the BERC 2018-2021 and ELKARTEK programs and through project KK-2018/00071; and by Spanish Ministry of Economy and Competitiveness MINECO through BCAM Severo Ochoa excellence accreditation SEV-2017-0718, and through project TIN2017-82626-R.

References

1. Bartlett, M.: An inverse matrix adjustment arising in discriminant analysis. Annals of Mathematical Statistics, vol. 22(1), pp. 107 – 111 (1951)
2. Batista, G.E.A.P.A., Prati, R.C., Monard, M.C.: A study of the behavior of several methods for balancing machine learning training data. ACM SIGKDD Explorations Newsletter, vol. 6(1), pp. 20 – 29 (2004)
3. Battiti, R.: Using mutual information for selecting features in supervised neural net learning. IEEE Transactions on Neural Networks, vol. 5(4), pp. 537 - 550 (1994)
4. Benkedjouh, T., Medjaher, K., Zerhouni, N., Rechak, S.: Remaining useful life estimation based on nonlinear feature reduction and support vector regression. Engineering Applications of Artificial Intelligence, vol. 26(7), pp. 1751 – 1760 (2013)
5. Box, G.E.P., Cox, D.R.: An analysis of transformations. Journal of the Royal Statistical Society, Series B, vol. 26 (2), pp. 211 - 252 (1964)
6. Bradley, A.P.: The use of the area under the ROC curve in the evaluation of machine learning algorithms. Pattern Recognition, vol. 30(7), pp. 1145 – 1159 (1997)
7. Branden, K.V., Hubert, M.: Robust classification in high dimensions based on the simca method. Chemometrics and Intelligent Laboratory Systems, vol. 79, pp. 10 – 21 (2005)
8. Bracewell, R.N.: The Fourier Transform and its Applications (third edition). McGraw-Hill Boston, ISBN 0-07-116043-4 (2000)
9. Breiman, L.: Bagging predictors. Machine Learning, vol. 24(2), pp. 123 – 140 (1996)
10. Breiman, L.: Random forests. Machine Learning, vol. 45(1), pp. 5 – 32 (2001)
11. Brown, G.: A New Perspective for Information Theoretic Feature Selection. Journal of Machine Learning Research, vol. 13, pp. 27 – 66 (2012)
12. Cabrera, D., Sancho, F., Sánchez, R.V., Zurita, G., Cerrada, M., Li, C., Vásquez, R.E.: Fault diagnosis of spur gearbox based on random forest and wavelet packet decomposition. Frontiers of Mechanical Engineering, vol. 10(3), pp. 277 – 286 (2015)
13. Cernuda, C., Lughofer, E., Märzinger, W., Kasberger, J.: NIR-based quantification of process parameters in polyetheracrylat (PEA) production using flexible non-linear fuzzy systems. Chemometrics and Intelligent Laboratory Systems, vol. 109(1), pp. 22 – 33 (2011)
14. Cernuda, C., Lughofer, E., Suppan, L., Röder, T., Schmuck, R., Hintenaus, P., Märzinger, W., Kasberger, J.: Evolving chemometric models for predicting dynamic process parameters in viscose production. Analytica Chimica Acta, vol. 725, pp. 22 – 38 (2012)
15. Cernuda, C., Lughofer, E., Hintenaus, P., Märzinger, Reischer, T., Pawliczek, M., W., Kasberger, J.: Hybrid adaptive calibration methods and ensemble strategy for prediction of cloud point in melamine resin production. Chemometrics and Intelligent Laboratory Systems, vol. 126, pp. 60 – 75 (2013)
16. Cernuda, C., Lughofer, E., Mayr, G., Röder, T., Hintenaus, P., Märzinger, W., Kasberger, J.: Incremental and decremental active learning for optimized self-adaptive calibration in viscose production. Chemometrics and Intelligent Laboratory Systems, vol. 138, pp. 14 – 29 (2014)
17. Cernuda, C., Lughofer, E., Klein, H., Forster, C., Pawliczek, M., Brandstetter, M.: Improved quantification of important beer quality parameters based on nonlinear calibration methods applied to FT-MIR spectra. Analytical and Bioanalytical Chemistry, vol. 409(3), pp. 841 – 857 (2017)
18. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: SMOTE: Synthetic Minority Oversampling TEchnique. Journal of Artificial Intelligence Research, vol. 16, pp. 321 - 357 (2002)
19. Chawla, N.V.: C4.5 and Imbalanced Data sets: Investigating the Effect of Sampling Method, Probabilistic Estimate, and Decision Tree Structure. Proceedings of the ICML'03 Workshop on Learning from Imbalanced Data sets, Washington, DC, USA (2003)
20. Chawla, N.V.: Data mining for imbalanced datasets: An overview. Maimon, O., Rokach, L. (editors), Data mining and knowledge discovery handbook (second edition), Springer, pp. 875 – 886 (2010)

21. Cho, K., Merriënboer, B., Gulcehre, C., Bougares, F., Schwenk, H., Bahdanau, D., Bengio, Y.: Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. Computer Research Repository (CoRR), vol. 1406.1078 (2014)
22. Chollet, F., et al.: Keras (2015). https://github.com/fchollet/keras
23. Chuang, A.: Time series analysis: Univariate and multivariate methods. Technometrics, vol. 33(1), pp. 108 - 109 (1991)
24. Cohen, L.: Time-Frequency Analysis. Prentice-Hall, New York (1995) ISBN 978-0135945322
25. Covell, M.M., Richardson, J.M.: A new, efficient structure for the short-time fourier transform, with an application in code-division sonar imaging. International Conference on Acoustics, Speech, and Signal Processing (ICASSP), vol. 3, pp. 2041 - 2044 (1991)
26. Daubechies, I.: Orthonormal bases of compactly supported wavelets. Communications on Pure and Applied Mathematics, vol. 41(7), pp. 909 - 996 (1988)
27. Drummond, C., Holte, R.: C4.5, class imbalance, and cost sensitivity: Why undersampling beats over-sampling. Proceedings of the ICML03 Workshop on Learning from Imbalanced Data Sets, Washington, DC, USA (2003)
28. Dudani, S.A.: The distance-weighted k-nearest neighbor rule. IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-6(4), pp. 325 – 327 (1976)
29. Duhamel, P., Vetterli, M.: Fast fourier transforms: a tutorial review and a state of the art. Signal Processing, vol. 19(4), pp. 259 - 299 (1990)
30. Fayyad, U.M., Irani, K.B.: Multi-interval discretization of continuous-valued attributes for classification learning. 13th International Joint Conference on Artificial Intelligence, pp. 1022 – 1027 (1993)
31. Ferri, C., Flach, P., Orallo, J., Lachice, N., (editors). ECAI2004 First Workshop on ROC Analysis in Artificial Intelligence (2004)
32. Fleuret, F.: Fast Binary Feature Selection with Conditional Mutual Information. The Journal of Machine Learning Research, vol. 5, pp. 1531 - 1555 (2004)
33. Freedman, D., Diaconis, P.: On the histogram as a density estimator: ℓ_2 theory. Probability Theory and Related Fields, vol. 57(4), pp. 453 - 476 (1981)
34. Friedman, N., Geiger, D., Goldszchmidt, M.: Bayesian network classifiers. Machine Learning, vol. 29(2-3), pp. 131 – 163 (1997)
35. Frigo, M., Johnson, S.G.: A Modified Split-Radix FFT With Fewer Arithmetic Operations. IEEE Transactions on Signal Processing, vol. 55(1), pp. 111 - 119 (2007)
36. García, S., Luengo, J., Sáez, J.A., López, V., Herrera, F.: A Survey of Discretization Techniques: Taxonomy and Empirical Analysis in Supervised Learning. IEEE Transactions on Knowledge and Data Engineering, vol. 25(4), pp. 734 – 750 (2013)
37. Garvey, D., Wigny, R.: PHM Data Challenge 2014. PHM Society. https://www.phmsociety.org/sites/phmsociety.org/files/PHM14DataChallenge.pdf
38. Gelper, S., Schettlinger, K., Croux, C., Gather, U.: Robust online scale estimation in time series: a model-free approach. Journal of Statistical Planning and Inference, vol. 139(2), pp. 335 - 349 (2008)
39. Gerretzen, J., Szymańska, E., Jansen, J., Bart, J., van Manen, H.-J., van den Heuvel, E.R., Buydens, L.: Simple and Effective Way for Data Preprocessing Selection Based on Design of Experiments. Analytical Chemistry. **87(24)**, 12096–12103 (2015)
40. Graves, A., Schmidhuber, J.: Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural Networks, vol. 18(5-6), pp. 602 – 610 (2005)
41. Greff, K., Srivastava, R.K., Koutník, J., Steunebrink, B.R., Schmidhuber, J.: LSTM: A Search Space Odyssey. IEEE Transactions on Neural Networks and Learning Systems, vol. 28(10), pp. 2222 – 2232 (2017)
42. Guha, S., Rastogi, R., Shim, K.: CURE: an efficient clustering algorithm for large databases. SIGMOD'98, Proceedings of the 1998 ACM SIGMOD international conference on management of data, pp. 73 – 84 (1998)
43. Guo, L., Ma, Y., Cukic, B., Singh, H.: Robust prediction of fault-proneness by random forests. 15th International Symposium on Software Reliability Engineering, pp. 417 – 428 (2004)

44. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. Journal of Machine Learning Research, 3(7-8), pp. 1157 - 1182 (2003)
45. Guyon, I., Elisseeff, A.: An Introduction to Feature Extraction. In: Guyon, I., Nikravesh, M., Gunn, S., Zadeh, L.A. (eds) Feature Extraction. Studies in Fuzziness and Soft Computing, vol 207, pp. 1 – 25. Springer, Berlin, Heidelberg (2006)
46. Hall, M.A.: Correlation-based feature selection for machine learning. PhD Thesis. University of Waikato, Hamilton, New Zealand (1999)
47. Hart, P.E.: The Condensed Nearest Neighbor Rule. IEEE Transactions on Information Theory, vol. 14, pp. 515 - 516 (1968)
48. Hastie, T., Tibshirani, R., Friedman, J.: Pathwise coordinate optimization, The Annals of Applied Statistics, vol. 1(2), pp. 302 – 332 (2007)
49. Hastie, T., Tibshirani, R., Friedman, J.: The elements of statistical learning: data mining, inference, and prediction (2nd Edition). Springer Series in Statistics, Springer (2009)
50. Hastie, T., Tibshirani, R., Friedman, J.: Regularized paths for generalized linear models via coordinate descent. Journal of Statistical Software, vol. 33(1), pp. 1 – 22 (2010)
51. He, H., Bai, Y., García, E.A., Li, S.: ADASYN: Adaptive synthetic sampling approach for imbalanced learning. IEEE International Joint Conference on Neural Networks, IEEE World Congress on Computational Intelligence, Hong Kong, pp. 1322 – 1328 (2008)
52. He, X., Niyogi, P.: Locality Preserving Projections. Proceedings of the 16th International Conference on Neural Information Processing Systems (NIPS'03), pp. 153 – 160 (2003)
53. Hinton, G., Roweis, S.: Stochastic Neighbor Embedding. Proceedings of the 15th Int. Conference on Neural Information Processing Systems (NIPS'02), pp. 857 – 864 (2002)
54. Hochreiter, S., Schmidhuber, J.: Long Short-Term Memory. Neural Computation, vol. 9(8), pp. 1735 – 1780 (1997)
55. Hofmann, T., Schölkopf, B., Smola, A.J.: Kernel methods in machine learning. The Annals of Statistics, vol. 36(3), pp. 1171 - 1220 (2009)
56. Huang, N.E., Shen, Z., Long, S.R., Wu, M.C., Shih, H.H., Zheng, Q., Yen, N.-C., Tung, C.C., Liu, H.H.: The empirical mode decomposition and the Hilbert spectrum for nonlinear and non-stationary time series analysis. Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences, vol. 454, pp. 903 - 995 (1998)
57. Hubert, M., Rousseeuw, P., Branden, K.V.: Robpca: A new approach to robust principal component analysis. Technometrics, vol. 47, pp. 64 – 79 (2005)
58. Japkowicz, N.:The Class Imbalance Problem: Significance and Strategies. Proceedings of the 2000 International Conference on Artificial Intelligence (IC-AI2000): Special Track on Inductive Learning, pp. 111 – 117, Las Vegas, Nevada, USA (2000)
59. Jo, T., Japkowicz, N.: Class imbalances versus small disjuncts. ACM SIGKDD Explorations Newsletter, vol. 6(1), pp. 40 – 49 (2004)
60. N. Johnson, S. Kotz, N. Balakrishnan, Continuous univariate distributions, no. 2 in Wiley series in probability and mathematical statistics: Applied probability and statistics, Wiley and Sons, 1995
61. Jolliffe, I.: Principal Components Analysis, Springer Verlag, Berlin Heidelberg New York (2002)
62. Jung, M., Niculita, O., Skaf, Z.: Comparison of Different Classification Algorithms for Fault Detection and Fault Isolation in Complex Systems. Procedia Manufacturing, vol. 19, pp. 111 – 118 (2018)
63. Kadambe, S., Boudreaux-Bartels, G.F.: A comparison of the existence of cross terms in the Wigner distribution and the squared magnitude of the wavelet transform and the short-time Fourier transform. IEEE Trans. on Signal Processing, vol. 40(10), pp. 2498 - 2517 (1992)
64. Kalchbrenner, N., Danihelka, I., Graves, A.: Grid Long Short-Term Memory. Computer Research Repository (CoRR), vol. 1507.01526 (2015)
65. Kohavi, R., John, G.H.: Wrappers for feature subset selection. Artificial Intelligence, vol. 97(1-2), pp. 273 – 324 (1997)
66. Kubat, M., Matwin, S.: Addressing the Curse of Imbalanced Training Sets: One Sided Selection. Proceedings of the Fourteenth International Conference on Machine Learning, pp. 179 - 186, Nashville, Tennesse. Morgan Kaufmann (1997)

67. Kwak, N., Choi, C.: Input Feature Selection for Classification Problems. IEEE Transactions on Neural Networks, vol. 13(1), pp. 143 - 159 (2002)
68. Laurikkala, J.: Improving Identification of Difficult Small Classes by Balancing Class Distribution. AIME'01, Proceedings of the 8th Conference on Artificial Intelligence in Medicine in Europe, pp. 63 – 66 (2001)
69. Li, D., Deogun, J., Spaulding, W., Shuart, B.: Towards missing data imputation — A study of fuzzy k-means clustering method. In: Tsumoto, S., Sowiski, R., Komorowski, J., Grzymaa-Busse, J. (eds.) Rough Sets and Current Trends in Computing (RSCTC 2004). Lecture Notes in Computer Science, vol. 3066, pp. 573-579. Springer Berlin Heidelberg (2004)
70. Li, M.: Fractal time series-a tutorial review. Mathematical Problems in Engineering, vol. 2010, pp. 1 – 26 (2010)
71. Lin, D., Tang, X.: Conditional Infomax Learning: An Integrated Framework for Feature Extraction and Fusion. In: Leonardis A., Bischof H., Pinz A. (eds) Computer Vision ECCV 2006. ECCV 2006. Lecture Notes in Computer Science, vol. 3951, pp. 68 – 82 (2006)
72. Ling, C., Li, C.: Data Mining for Direct Marketing Problems and Solutions. Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98), New York, NY. AAAI Press, pp. 73 – 79 (1998)
73. Loutas, T., Roulias, D., Georgoulas, G.: Remaining Useful Life Estimation in Rolling Bearings Utilizing Data-Driven Probabilistic ε-Support Vectors Regression. IEEE Transactions on Reliability, vol. 62(4), pp. 821 – 832 (2013)
74. Lughofer, E.: FLEXFIS: A robust incremental learning approach for evolving Takagi-Sugeno fuzzy models. IEEE Transactions on Fuzzy Systems, vol. 16(6), pp. 1393 – 1410 (2008)
75. Maaten, L., Hinton, G.: Visualizing Data using t-SNE. Journal of Machine Learning Research, vol. 9, pp. 2579 – 2605 (2008)
76. Maesschalck, R.D., Candolfi, A., Massart, D., Heuerding, S.: Decision criteria for soft independent modelling of class analogy applied to near infrared data. Chemometrics and Intelligent Laboratory Systems, vol. 47, pp. 65 – 77 (1999)
77. Mahalanobis, P.: On the generalised distance in Statistics. Proceedings of the National Institute of Sciences of India, vol. 2(1), pp. 49 – 55 (1936)
78. Mallat, S.G.: A theory for multiresolution signal decomposition: the wavelet representation. IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 11(7), pp. 674 - 693 (1989)
79. Maloof, M.: Learning when data sets are imbalanced and when costs are unequal and unknown. Proceedings of the ICML03 Workshop on Learning from Imbalanced Data Sets, Washington, DC, USA (2003)
80. Mann, H.B., Whitney, D.R.: On a Test of Whether one of Two Random Variables is Stochastically Larger than the Other. Annals of Mathematical Statistics, vol. 18(1), pp. 50 - 60 (1947)
81. Nikzad-Langerodi, R., Lughofer, E., Cernuda, C., Reischer, T., Kantner, W., Pawliczek, M., Brandstetter, M.: Calibration model maintenance in melamine resin production: Integrating drift detection, smart sample selection and model adaptation. Analytica Chimica Acta, vol. 1013, pp. 1 – 12 (2018)
82. Nunkesser, R., Fried, R., Schettlinger, K., Gather U.: Online analysis of time series by the Q_n estimator. Computational Statistics and Data Analysis, vol. 53(6), pp. 2354 - 2362 (2009)
83. Oba, S., Sato, M., Takemasa, I., Monden, M., Matsubara, K., et al.: A Bayesian missing value estimation method for gene expression profile data. Bioinformatics **19**, 2088–2096 (2003)
84. Oliveira, M.A., Araujo, N.V.S., Silva, R.N., Silva, T.I., Epaarachchi, J.: Use of Savitzky-Golay Filter for Performances Improvement of SHM Systems Based on Neural Networks and Distributed PZT Sensors. Sensors, vol. 18(1), no.152 (2018)
85. Pedrycz, W., Gomide, F.: Fuzzy Systems Engineering: Toward Human-Centric Computing. John Wiley and Sons, Hoboken, New Jersey (2007)
86. Peng, H., Long, F., Ding, C.: Feature Selection Based on Mutual Information: Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27(8), pp. 1226 - 1238 (2005)
87. Phua, C., Alahakoon, D.: Minority report in fraud detection: Classification of skewed data. ACM SIGKDD Explorations Newsletter, vol. 6(1), pp. 50 – 59 (2004)

88. Propes, N.C., Rosca, J.: PHM Data Challenge 2016. PHM Society. https://www.phmsociety.org/sites/phmsociety.org/files/PHM16DataChallengeCFP.pdf
89. Qiu, G.: An improved recursive median filtering scheme for image processing. IEEE Transactions on Image Processing, vol. 5(4), pp. 646 - 648 (1996)
90. Rezgui, W., Mouss, N.K., Mouss, L.H., Mouss, M.D., Benbouzid, M.: A regression algorithm for the smart prognosis of a reversed polarity fault in a photovoltaic generator. 2014 International Conference on Green Energy, pp. 134 – 138 (2014)
91. Roweis, S., Saul, L.: Nonlinear dimensionality reduction by locally linear embedding. Science, vol. 290(5500), pp. 2323 - 2326 (2000)
92. Rubin, D.B.: Multiple imputation for nonresponse in survey, 1. John Wiley and Sons Inc. (2008)
93. Saeys, Y., Inza, I., Larrañaga, P.: A review of feature selection techniques in bioinformatics. Bioinformatics, vol. 23(19), pp. 2507 – 2517 (2007)
94. Said E. Said and David A. Dickey. Testing for unit roots in autoregressive-moving average models of unknown order. Biometrika, 71(3):599607, 1984
95. Sakia, R.M.:, The Box-Cox transformation technique: a review, The Statistician, vol. 41(2), pp. 169 - 178 (1992)
96. Savitzky, A., Golay, M.J.E.: Smoothing and Differentiation of Data by Simplified Least Squares Procedures. Analytical Chemistry, vol. 36(8), pp. 1627 - 1639 (1964)
97. Schölkopf, B., Smola, A., Müller, K.R.: Kernel principal component analysis. In: Gerstner W., Germond A., Hasler M., Nicoud JD. (eds) Artificial Neural Networks ICANN'97. Lecture Notes in Computer Science, vol 1327. Springer, Berlin, Heidelberg (1997)
98. Schölkopf, B., Smola, A.J.: Learning with kernels - Suport vector machines, regularization, optimization and beyond. MIT Press, London (2002)
99. Serdio, F., Lughofer, E., Pichler, K., Buchegger, T., Pichler, M., Efendic, H.: Multivariate Fault Detection using Vector Autoregressive Moving Average and Orthogonal Transformation in Residual Space. 2013 Annual Conference of the Prognostics and Health Management (PHM) Society, New Orleans, LA, USA, pp. 1 – 8 (2013)
100. Serdio, F., Lughofer, E., Pichler, K., Buchegger, T., Efendic, H.: Residual-based fault detection using soft computing techniques for condition monitoring at rolling mills. Information Sciences, vol. 259, pp. 304 – 320 (2014)
101. Serdio, F., Lughofer, E., Pichler, K., Buchegger, T., Pichler, M., Efendic, H.: Fault detection in multi-sensor networks based on multivariate time-series models and orthogonal transformations. Information Fusion, vol. 20, pp. 272 – 291 (2014)
102. Serdio, F., Lughofer, E., Zavoianu, A.C., Pichler, K., Buchegger, T., Pichler, M., Efendic, H.: Improved fault detection employing hybrid memetic fuzzy modeling and adaptive filters. Applied Soft Computing, vol. 51, pp. 60 – 82 (2017)
103. Shannon, C.E.: A mathematical theory of communication. The Bell System Technical Journal, vol. 27(3), pp. 379 – 423 (1948)
104. Sharpley, R.C., Vatchev, V.: Analysis of the intrinsic mode functions. Constructive Approximation, vol. 24(1), pp. 17 - 47 (2006)
105. Silverman, B.W., Jones, M.C.: An Important Contribution to Nonparametric Discriminant Analysis and Density Estimation: Commentary on Fix and Hodges (1951). International Statistical Review / Revue Internationale De Statistique, vol. 57(3), pp. 233 – 238 (1989)
106. Smith, M.R., Martínez, T., Giraud-Carrier, C.: An Instance Level Analysis of Data Complexity. Machine Learning, vol. 95, no. 2, pp. 225–256 (2014)
107. Smola, A.J., Schölkopf, B.: A tutorial on support vector regression. Statistics and Computing, vol. 14, pp. 199 - 222 (2004)
108. Solberg, A. H., Solberg, R.: A Large-Scale Evaluation of Features for Automatic Detection of Oil Spills in ERS SAR Images. International Geoscience and Remote Sensing Symposium, pp. 1484 - 1486 (1996)
109. Tan, L., Jiang, J.: Digital signal processing: fundamentals and applications (second edition). Academic Press, Elsevier 2013
110. Tenenbaum, J.B., Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science, vol. 290(5500), pp. 2319 – 2323 (2000)

111. Tibshirani, R.: Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society, vol. B 58, pp. 267 – 288 (1996)
112. Tomek, I.: Two Modifications of CNN. IEEE Transactions on Systems, Man and Cybernetics, vol. 6, pp. 769 - 772 (1976)
113. Troyanskaya, O., Cantor, M., Sherlock, G, Brown, P., Hastie, T., et al.: Missing value estimation methods for dna microarrays. Bioinformatics. **17**, 520–525 (2001)
114. Tschumitschew, K., Klawonn, F.: Incremental quantile estimation. Evolving Systems, vol. 1(4), pp. 253 – 264 (2010)
115. Vapnik, V: Statistical learning theory. Wiley, New York (1998)
116. Varmuza, K., Filzmoser, P.: Introduction to Multivariate Statistical Analysis in Chemometrics. CRC Press, Boca Raton (2009)
117. Vidal-Naquet, M., Ullman, S.: Object recognition with informative features and linear classification. 9th IEEE Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 281 – 288 (2003)
118. Ville, J.: Thorie et Applications de la Notion de Signal Analytique. Cbles et Transmission, vol. 2, pp. 61 - 74 (1948)
119. Wang, C., Zhang, Y., Zhong, Z.: Fault diagnosis for diesel valve trains based on timefrequency images. Mechanical Systems and Signal Processing, vol. 22(8), pp. 1981 - 1993 (2008)
120. Weaver, H.J.: Applications of discrete and continuous Fourier analysis. Wiley-Interscience (1983)
121. Weiss, G., Provost, F.: Learning when Training Data are Costly: The Effect of Class Distribution on Tree Induction. Journal of Artificial Intelligence Research, vol. 19, pp. 315 - 354 (2003)
122. Welch, P.: The use of fast fourier transform for the estimation of power spectra: a method based on time averaging over short, modified periodograms. IEEE Transactions on Audio and Electroacoustics, vol. 15(2), pp. 70 - 73 (1967)
123. Wilson, D.L.: Asymptotic Properties of Nearest Neighbor Rules Using Edited Data. IEEE Transactions on Systems, Man, and Cybernetics, vol. SMC-2, no. 3, pp. 408-421, (1972)
124. Wolpert, D., Macready, W.: No free lunch theorems for optimization, IEEE Transactions on Evolutionary Computation, vol. 1(1), pp. 67 – 82 (1997)
125. Wu, D., Jennings, C., Terpenny, J., Gao, R., Kumara, S.: A Comparative Study on Machine Learning Algorithms for Smart Manufacturing: Tool Wear Prediction Using Random Forests. Journal of Manufacturing Science and Engineering, vol. 139(7), no. 071018 (2017)
126. Wu, T.Y., Chen, J., Wang, C.X.: Characterization of gear faults in variable rotating speed using Hilbert-Huang transform and instantaneous dimensionless frequency normalization. Mechanical Systems and Signal Processing, vol. 30, pp. 103 - 122 (2012)
127. Yang, B.S., Di, X., Han, T.: Random forests classifier for machine fault diagnosis. Journal of Mechanical Science and Technology, vol. 22, pp. 1716 – 1725 (2008)
128. Yang, H., Moody, J.: Data Visualization and Feature Selection: New Algorithms for Non-gaussian Data. Advances in Neural Information Processing Systems, vol. 12, pp. 687 – 693 (1999)
129. Yao, K., Cohn, T. Vylomova, K., Duh, K., Dyer, C.: Depth-Gated Long Short-Term Memory. Computer Research Repository (CoRR), vol. 1508.03790 (2015)
130. Zavoianu, A.C., Lughofer, E., Bramerdorfer, G., Amrhein, W., Klement, E.P.: An Effective Ensemble-Based Method for Creating On-the-Fly Surrogate Fitness Functions for Multi-Objective Evolutionary Algorithms. International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2013), pp. 235 – 242 (2013)
131. Zhang, J., Mani, I.: kNN approach to unbalanced data distributions: A case study involving information extraction. Proceedings of the ICML2003 workshop on learning from imbalanced datasets, Washington, DC, USA (2003)
132. Zhang, L., Xiong, G., Liu, H., Zou, H., Guo, W.: Bearing fault diagnosis using multi-scale entropy and adaptive neuro-fuzzy inference. Expert Systems with Applications, vol. 37(8), pp. 6077 - 6085 (2010)
133. Zou, H. Hastie, T.: Regularization and variable selection via the elastic net, Journal of the Royal Statistical Society, vol. 67(2), pp. 301 – 320 (2005)