Array 7 (2020) 100034

Contents lists available at ScienceDirect

Array

journal homepage: www.elsevier.com/journals/array/2590-0056/open-access-journal

Process-monitoring-for-quality — A machine learning-based modeling for rare event detection

Carlos A. Escobar [a,*], Ruben Morales-Menendez [b], Daniela Macias [b]

[a] Global Research & Development, General Motors, Warren, MI, USA
[b] Tecnológico de Monterrey, School of Engineering and Sciences, Monterrey NL, Mexico

ARTICLE INFO

Index Terms:
Quality control
Manufacturing systems
Machine learning
Feature elimination
Model selection
Unbalanced binary data
Defect detection

ABSTRACT

Process Monitoring for Quality is a *Big Data*-driven quality philosophy aimed at defect detection through binary classification and empirical knowledge discovery. It is founded on *Big Models*, a predictive modeling paradigm that applies *Machine Learning*, statistics and optimization techniques to process data to create a manufacturing functional model. Functional refers to a parsimonious model with high detection ability that can be trusted by engineers, and deployed to control production. A parsimonious modeling scheme is presented aimed at rare quality event detection, parsimony is induced through feature selection and model selection. Its unique ability to deal with highly/ultra-unbalanced data structures and diverse learning algorithms is validated with four case studies, using the Support *Vector Machine*, *Logistic Regression*, *Naive Bayes* and *k-Nearest Neighbors* learning algorithms. And according to experimental results, the proposed learning scheme significantly outperformed typical learning approaches based on the l_1-regularized logistic regression and *Random Undersampling Boosting* learning algorithms, with respect to parsimony and prediction.

1. Introduction

Because of ever increasing customer demands, manufacturers are in a constant competition for improving quality and reliability in products. These attributes are achieved by correct execution of the manufacturing process and by an effective process monitoring system.

Several researchers have worked on this problem. A framework for multiple release problems using a two-step fault detection procedure and fault removal process was proposed by Ref. [1]. A state-of-the-art report of the most important papers in this domain is presented in Refs. [2]. A similar project, but based on Bayesian Nets was suggested by Ref. [3]. A data mining approach for defect analysis and prevention of industrial products in Ref. [4].

Process Monitoring for Quality (PMQ) is a *Big Data*-driven quality philosophy aimed at defect detection through binary classification (good/bad) and empirical knowledge discovery through feature/model interpretation [5]. It is a blend of process monitoring and quality control founded on *Big Models (BM)*; a predictive modeling paradigm that uses *Machine Learning* (*ML*), statistics and optimization techniques, Fig. 1, to develop a manufacturing functional model: *final model* (classifier). Functional, refers to a parsimonious classifier with a high detection capacity; parsimony facilitates information extraction, induces model trust [6] and promotes explainability [7]; desired characteristics for a classifier to be deployed into production, Fig. 2. *PMQ* has the potential to solve engineering intractable problems e.g., detecting defects that are not detected by *Statistical Process Control* methods.

Constant product innovation forces manufacturing engineers to launch production systems even without a comprehensive understanding of the process. Therefore, the huge amount of process data (e.g., signals) is used to create hundreds or even thousands of features i.e., hyper-dimensional feature spaces. Which frequently include irrelevant and redundant ones [8,9] that tend to hamper the learning process [10].

Since most manufacturing companies generate only a few *Defects Per Million of Opportunities (DPMO)*, rare quality event detection is one of the modern intellectual challenges posed to this industry. From *ML* perspective, manufacturing-derived data sets for binary classification of quality tend to be highly/ultra-unbalanced (minority class count $< 1\%$). The problem with these data structures is that the learning algorithms misclassify most of the minority class as the majority class, e.g., fail to detect. Since it is harder for the algorithm to learn the minority class.

The unbalanced classification problem is taking a lot of attention from the *ML* community [11,12]. Extensive efforts and significant progress have been made in recent years to address this intellectual challenge. The main research efforts are broken down in five categories: (1)

* Corresponding author.
E-mail address: carlos.1.escobar@gm.com (C.A. Escobar).

https://doi.org/10.1016/j.array.2020.100034
Received 24 January 2020; Received in revised form 31 May 2020; Accepted 1 July 2020
Available online 6 August 2020

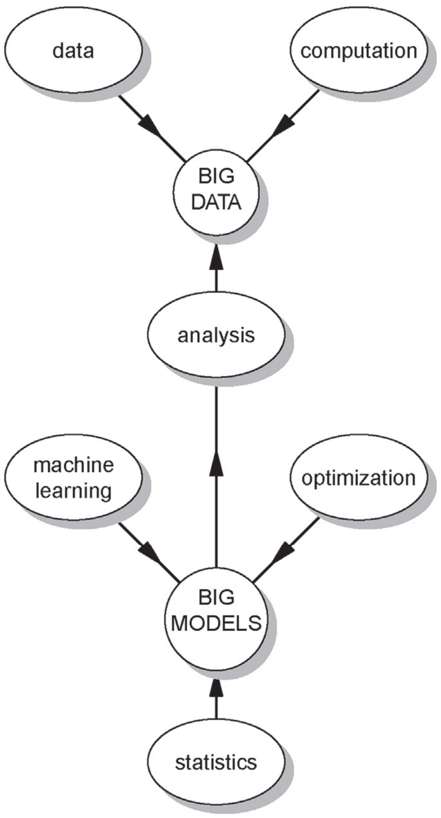

Fig. 1. Big data – big models.

over/under-sampling methods, (2) cost sensitive learning, (3) kernel-based learning, (4) active learning and (5) novelty detection. Where oftentimes the inherent ad hoc data manipulation (e.g., over/-under sampling, miss-classification costs, using small pools of data) approach lacks of theoretical foundation and principles to guide the development of systematic methods that can be efficiently generalized. In this context, learning from the original data set is identified as further research work [12]. Basically, the authors encourage the research community to investigate the development of algorithms/methods that could learn from whatever highly/ultra unbalanced data is presented without manipulations. Comprehensive reviews of unbalanced learning are presented in Ref. [12,13].

The feature explosion (hyperdimensional feature spaces) combined with high conformance production rates (unbalanced binary data) are two of the most important challenges of *Big Data* initiatives in manufacturing that inspired the development of *PMQ-Learning (PMQ-L)*. The *Hybrid Feature Selection and Pattern Recognition (HFSPR)* approach proposed in this paper with the capacity to effectively learn from the original data set and to identify the driving features.

The proposal is a novel parsimonious modeling scheme, that can be applied to train the *Naive Bayes* (*NB*), Support *Vector Machine* (*SVM*), *K-Nearest Neighbor* (*KNN*), *Logistic Regression* (*LR*) and *Fisher Linear Discriminant* (*FLD*) learning algorithms. The goal is a rare quality event detection through parsimonious modeling, where parsimony is induced through *Feature Selection (FS)* and *Model Selection (MS)*.

The proposed scheme combines the *Hybrid Correlation and Ranking-based* (*HCR*) and *ReliefF* filtering algorithms to select the most relevant features. To boost parsimony, a set of nested *Candidate Models (CM)* is developed and then, the *Penalized Maximum Probability of Correct Decision (PMPCD) MS* criterion is applied to select the *final model*. It can be virtually applied to any learning algorithm in which complexity is defined by the number of features in the model. Its universal applicability is demonstrated by analyzing four highly/ultra-unbalanced data sets using different learning algorithms. Empirical results demonstrate its capacity of finding a good quality solution after creating a few *CM*.

This paper is organized as follows: A review of the theoretical background is in section 2. Section 3 describes the *PMQ-L* framework, followed by four binary classification empirical studies in section 4. A comparative analysis is given in Section 5. Finally, section 6 concludes the research.

2. Theoretical background

2.1. FS methods

FS is the procedure of choosing a subset of good features by eliminating irrelevant and redundant ones. From a given data set, evaluating all possible combinations (2^n) becomes a NP-hard problem as the number of features grow up [14]. The advantages of *FS* methods are: (1) enable the learning algorithm to train faster, (2) reduce over-complexity of a model making it easier to understand/interpret, (3) improve generalization (prediction on unseen data) if the right subset is chosen, and (4)

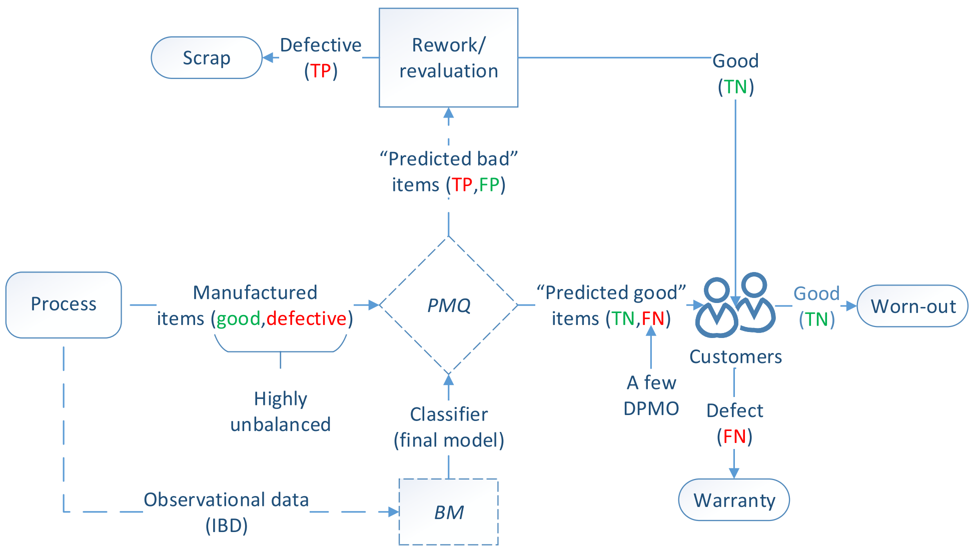

Fig. 2. *PMQ*-based quality control.

Table 1
Confusion matrix.

	Predicted good	Predicted bad
Good item	True Negative (TN)	False Positive (FP)
Bad item	False Negative (FN)	True Positive (TP)

prevent over-fitting. The *FS* methods broadly fall into 3 classes: filters, wrappers, and embedded [15].

Filter (preprocessing) methods are applied before the learning process to eliminate irrelevant/redundant features. Relevant features are selected using a predefined fitness function, which can be based on dependency, distance, consistency or discriminative capacity [16]. Computed scores are used to determine the fitness of each feature and are compared with a relevance threshold to select a subset of relevant features [10]. These methods select features independently of the learning algorithm.

In contrast, *wrappers* methods use the learning algorithm as a black-box to evaluate the relative performance of a feature subset [17,18]. In this procedure, a set of candidate features are input to the learning algorithm, and the prediction performance is used as the objective function to evaluate the feature subset. Although *wrapper* methods tend to be computationally intensive, they perform better than *filters*, due to the bias induced by the algorithm [15].

In *embedded* methods, the *FS* task is integrated as part of the learning process. The *LASSO* with the L_1 penalty and *Ridge* with the L_2 penalty are the most common approaches in this category. They shrink irrelevant and trivial features to zero or almost zero respectively [19,20].

Hybrid approaches have been proposed to take advantage of the particular characteristics of each method [21]. These approaches mainly focus on combining *filter* algorithms with either *wrappers* or regularization to solve the scalability problem, induce parsimony, and to achieve the best possible learning performance. The basic idea is to break down the *FS* problem into several stages, namely feature ranking, correlation-based feature elimination, and prediction optimization.

2.2. ReliefF

ReliefF ranks features according to their discriminative capacity [22]. It searches for k number of nearest neighbors of the same class (*hits*), as well as of the different class (*misses*) to evaluate the fitness of each feature. This procedure is repeated m times, which is the number of randomly selected instances. Features are weighted and ranked based on the average of the distances of all *hits* and all *misses* [23,24]. k is a hyperparameter (user-specified) that provides protection to the effect of noise and controls the locality of the estimates. Once all features have been ranked, they are selected based on τ, a significance threshold proposed by Ref. [22]. Features with an estimated weight below τ are considered irrelevant and therefore eliminated. The proposed limits for τ are $0 < \tau \leq 1/\sqrt{\alpha m}$ [23]; where α is the probability of accepting an irrelevant feature as relevant. *ReliefF* does not eliminate redundant features.

2.3. Correlation-based redundancy measure

The *Pearson* product-moment correlation coefficient (r_{xy}) is used as a measure of redundancy between two random variables [25]. It is a measure of strength of linear relationship between two variables (x, y), and it can take a range of values $[-1, 1]$. A value of 0 indicates there is no linear relationship, while an absolute value of 1 (or close to 1) indicates strong linear relationship, and therefore considered highly redundant.

2.4. Maximum probability of Correct Decision — A measure of prediction performance

In the context of binary classification, a positive label refers to a defective (bad) item, whereas a negative label refers to a good quality item. The prediction performance of a classifier is summarized in a confusion matrix [26]. This table is used to contrast predicted labels with the real quality characteristic, Table 1, and to compute relevant measures of classification performance.

A type-I error (α) is compared with a *FP* prediction; a type-II (β) error is compared with a *FN* [25]:

$$\alpha = \frac{\text{FP}}{\text{FP} + \text{TN}}, \quad \beta = \frac{\text{FN}}{\text{FN} + \text{TP}} \tag{1}$$

The *MPCD* is a probabilistic measure of classification performance that is driven by detection. Since it is very sensitive to *FN* (missing defective items) in highly/ultra-unbalanced classes [5]. The α and β errors are combined to estimate its score:

$$\text{MPCD} = (1-\alpha)(1-\beta) \tag{2}$$

where higher score indicates better classification, *MPCD* $\in [0, 1]$.

2.5. Penalized Maximum Probability of Correct Decision

It is a *MS* criterion based on *MPCD* that efficiently solves the posed tradeoff between model complexity and prediction ability. The basic idea of the criterion is to induce parsimony by penalizing for extra features in the model. The formulation includes a rewarding term based on prediction/detection $(1-\alpha)(1-\beta)$ and a light penalization term $-ln(K)/34.55$ based on the number of features (K) in the model. And hence, *CM* with extra features with negligible contribution to prediction will have a smaller score (and therefore never selected). It is designed to be applied to *ML*-based models in which their complexity can be defined by the number of features, e.g., *SVM*, *LR*, *NB*, *KNN* and *FLD*. Insights about the development and properties of this criterion can be found in Ref. [27].

$$PMPCD = (1-\alpha)(1-\beta) - ln(K)\,/\,34.55 \tag{3}$$

The model with the highest estimated value on the validation set [28,29] is the preferred one.

2.6. Hybrid Correlation and Ranking-based (HCR) algorithm

The *HCR* algorithm [30] eliminates redundant features based on *Pearson*'s correlation coefficients and the *ReliefF*-based ranking. Features are eliminated if their correlation score is greater than δ, a user-specified threshold to determine if the discriminative information between the compared features is redundant. The basic idea of the algorithm is to keep the *best* feature – highest ranked – from a set of two or more highly correlated variables.

2.7. PMQ-based quality control

Rare quality event detection is one of the main applications of *PMQ*.

Table 2
Confusion matrix.

4). *Solution, evaluation and discussion*: Although the feature combination is subject to combinatorial explosion, 1.8×10^{16} of combinations, the *PMQ-L* approach only required 83 models to find a solution. To evaluate its relative quality, an exhaustive search was performed with all the possible combinations – up to two features – and compared with the *final model.* Since no *MS* is performed, the training set is used to develop the models and the testing set to evaluate their generalization ability: (1) $54(C_1)$ 1-feature models, Fig. 7(top); and (2) $1431(C_2)$ 2-feature models, Fig. 7(bottom).

	Predicted good	Predicted bad
Good item	9488	5
Bad item	0	7

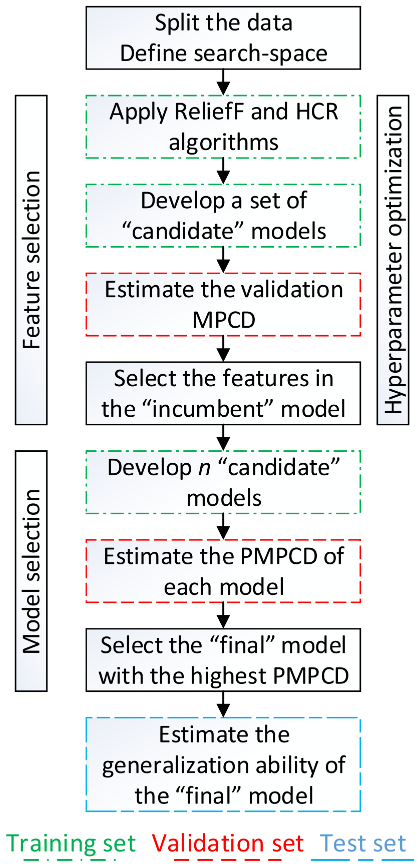
Split the data
Define search-space
Apply ReliefF and HCR algorithms
Develop a set of "candidate" models
Estimate the validation MPCD
Select the features in the "incumbent" model
Feature selection
Hyperparameter optimization
Develop *n* "candidate" models
Estimate the PMPCD of each model
Select the "final" model with the highest PMPCD
Model selection
Estimate the generalization ability of the "final" model
Training set
Validation set
Test set

Fig. 3. *PMQ-L* framework.

As depicted in Fig. 2, a typical manufacturing process generates only a few *DPMO*. The *BM* learning paradigm is applied to process data to design a classifier with high detection capacity to be deployed at the plant, e.g., *final model.* Since prediction is performed under uncertainty, a classifier can commit *FP* and *FN* (i.e., fail to detect) errors. Whereas a good item predicted as bad (*FP*) would not generate a critical problem, since in a second level inspection/revaluation would be back in the value-adding process, a *FN* would become a warranty event. And if this"miss" is a critical component/device, then it could have a serious economic impact as well as on the company's reputation.

3. PMQ-learning — a Hybrid Feature Selection and Pattern Recognition approach

A new parsimonious modeling method is presented, it is a flexible approach with a unique ability to deal with highly/ultra-unbalanced data structures and diverse learning algorithms to model linear and no-linear patterns.

Parsimonious modeling is induced through *FS* and *MS*, Fig. 3. Since most manufacturing systems are time-dependent, cross-validation methods are not encouraged. Instead, time-ordered hold-out method seems to be more appropriate. The data set should be partitioned into three subsets (i.e., training, validation, testing) [29]. And the search space is defined by many candidate pairwise combinations – based on different values of k for *ReliefF* and δ for *HCR*. The values of k can be determined by generating a logarithmically spaced vector [31] e.g., p logarithmically spaced points between decades $[10^a, 10^b]$, where $X = sum(bad)$ in the training set, $a = 0$ and $b = log_{10}(X)$.

1) *Feature Selection (FS):* The primary purpose of this step, Fig. 3, is to find a small subset of relevant features with high prediction capacity. Since the optimal combination – with respect to prediction – of k and δ is not known in advance, a hyperparameter optimization [32], is performed through a grid search [33,34]. Using the training set, irrelevant and redundant features are eliminated by applying *ReliefF* and *HCR* algorithms. Features are ranked based on *ReliefF* and irrelevant features are eliminated based on τ – significance threshold. From the selected features, high correlations are eliminated based on δ. These two steps are performed in a filter-type approach, where the learning algorithm is not considered.

A *CM* is developed with the subset of features at each pairwise combination, and the predictive fitness of each model is evaluated to find the *incumbent* (best) model – highest validation *MPCD*. The features in the *incumbent* model are selected and their associated *ReliefF* ranking recorded.

2) *Model Selection (MS)*: Although a good feature subset has been obtained in the *FS* step, Fig. 3, their individual relevance in the model is not known. To evaluate their prediction-contribution, a set of n nested *CM* is developed – n is the number of selected features – using the top 1 feature in the first *CM*, the top 2 features in the second one, and so on. Finally, the *PMPCD* of each *CM* is estimated and used as a *MS* criterion to induce parsimony – solve the tradeoff between model complexity and prediction ability. The *final model* is the one with the highest *PMPCD* score.
3) *Generalization evaluation*: To obtain an unbiased estimation (or closest to) of the generalization ability of the *final model*, the prediction on testing set (unseen data) should be reported in a confusion matrix, last step Fig. 3.

The outcome of this method is a parsimonious classifier with high detection ability, as the analytical tools used are aimed at analyzing highly/ultra-unbalanced data structures. Parsimony does not only improve the learning ability, but also helps to identify the few driving features of the system.

In concordance with *PMPCD*, the application of the proposed learning scheme is limited to classifiers in which their complexity is mainly defined by the number of features in the model, e.g., *SVM*, *LR*, *NB*, *KNN* and *FLD*. Therefore, learning algorithms such as neural networks, random forest, etc. are out of the scope.

4. Case studies

Four highly/ultra-unbalanced data sets were analyzed using the *SVM*, *LR*, *NB*, and *KNN* learning algorithms [35]. First, a full analysis is presented using the *NB* algorithm on a manufacturing-derived data set. Then, the same procedure was applied to three different data sets. Due to space limitations, only results were reported.

4.1. Case study 1

The data used for this analysis[1] was collected from the *Ultrasonic Metal Welding* of battery tabs for the *Chevrolet Volt* [5], a well-controlled process that only generates a few *DPMO*. It contains 54 features with a binary outcome (*good vs bad* quality), its data structure is ultra-unbalanced – 35 *bad* welds out of 40,231 examples (0.09%). Three data sets are created following a time-ordered hold-out validation approach: training set (18,495, including 20 *bad*), validation set (12,236 - 8 *bad*), testing set (9500 - 7 *bad*).

[1] Privately stored in the Manufacturing Research Lab of General Motors.

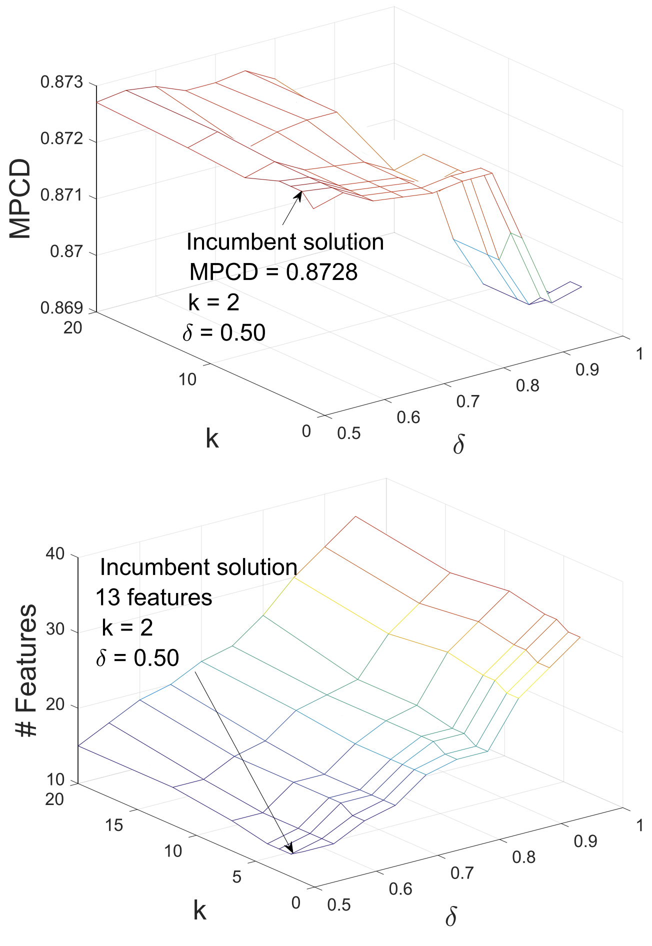

Fig. 4. *CM* information (denoted by line intersections).

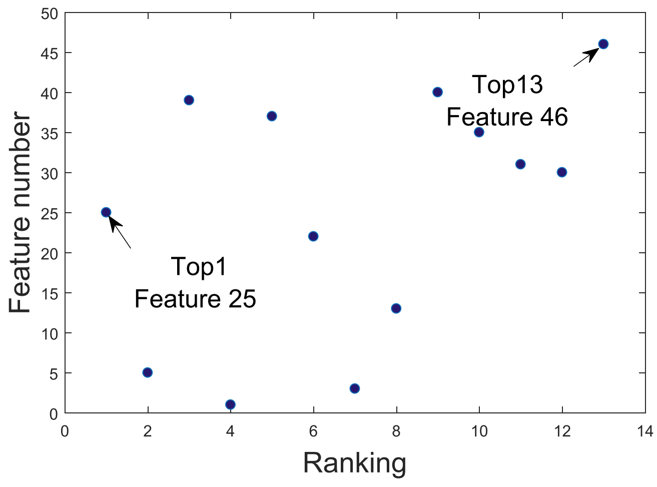

Fig. 5. Features in the *incumbent* model.

1) *Feature Selection (FS)*: The search space contains 70 pairwise combinations; for *ReliefF*, 7 logarithmically spaced points were defined – $k = \{1, 2, 3, 4, 7, 12, 20\}$ – and for δ, 10 even spaced points – $\delta = \{0.50, 0.55, \ldots, 0.95\}$. At each combination, feature relevance was determined by comparing their weights with $\tau = 0.0329$ – calculated with an α of 0.05, and *m* of 18,495. Fig. 4 shows prediction results (validation *MPCD*) and number of features of each *CM*.

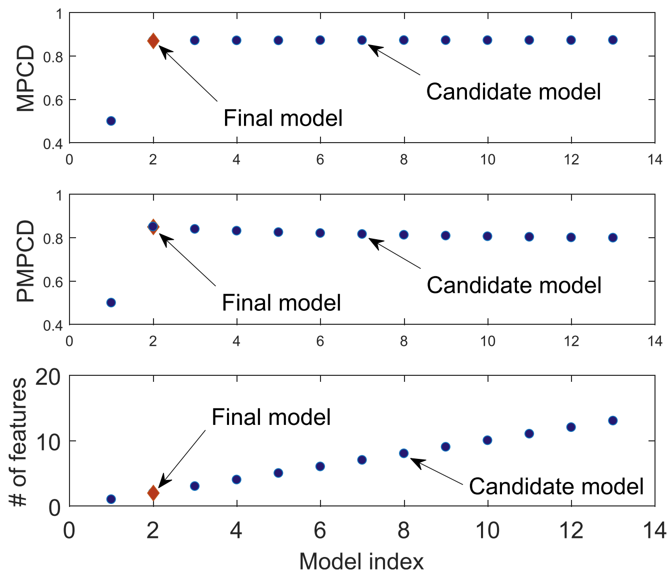

Fig. 6. *CM* using the top 13 features.

3). *Generalization evaluation*: The testing set (9500 - including 7 *bad*) was used to estimate the unbiased generalization ability of the *final model*, recognition rates are summarized in the confusion matrix, Table 2. This model includes only two features (25, 5), and it correctly detected the seven defective items with only five *FPs* – $MPCD = 0.9995$. It is clear that the system can be justified by only these two features.

Table 3
Top models (**PMQ-L* solution).

Model index	Features	MPCD	FN
1032	26, 33	0.9998	2
1035	26, 36	0.9998	2
413	9, 26	0.9997	3
1042	26, 43	0.9997	3
1044	26, 45	0.9997	3
1045	26, 46	0.9996	4
Final	5, 25	0.9995	5*

According to the grid search results, the *incumbent* model has an estimated validation $MPCD = 0.8728$, Fig. 4(top), and 13 features, Fig. 4(bottom). This model was developed with these relevant hyperparameters: $k = 2, \tau = 0.0329, \delta = 0.50$. All *CM* failed to detect one of the defective items; therefore, the $\beta = 0.125$ in all models. And they are basically competing over the α error. As displayed by the plots, as the number of low quality features included in the model increases, the α error increases too. The proposed hyperparameter optimization allowed to find a good subset of features.

2) *Model Selection (MS)*: To induce parsimony, 13 CM were created, and *PMPCD* was used as a *MS* criterion to find the *final model.* The basic idea is to evaluate the individual prediction-contribution of each of the 13 selected features, Fig. 5 shows the selected features and their associated ranking. *CM*-1 contains top-1 feature (25), *CM*-2 contains the top-2 features (25, 5) and so on.

According to the *MS* criterion, *CM*-2 should be selected, with an estimated $PMPCD = 0.8501$, Fig. 6. This analysis, discloses that only two features are needed to approximate the pattern in the manufacturing system, since the prediction improvement is not significant if more features are added to the *final model.*

Based on exhaustive search, no single-feature model has better generalization ability. Whereas six 2-feature models outperformed the *final model*, Table 3 summarizes their relevant information. However,

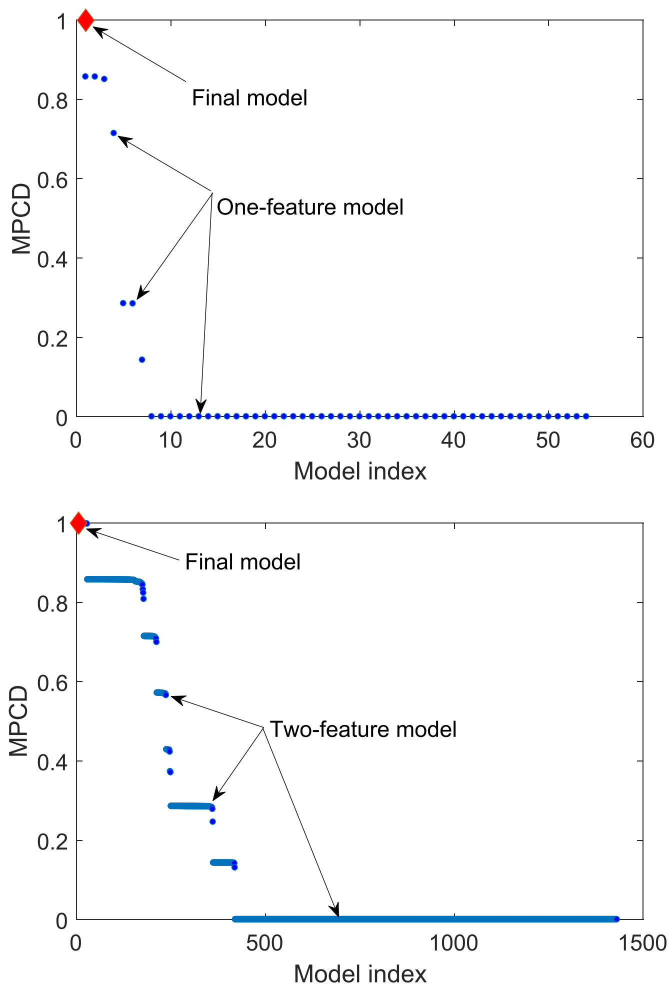

Fig. 7. MPCD exhaustive search in the 1-feature and 2-feature spaces.

evaluating all possible combinations to find an optimal solution rapidly becomes unfeasible as the feature space grows up.

The optimal solution could be defined as the model with the least number of features and the highest prediction ability. In this case study, if there is no other model with an estimated *MPCD* > 0.9998, the optimal solutions would be model indexes 1032 and 1035 Table 3. However, since the number of combinations is huge, a model with more features may have greater *MPCD*. Oftentimes due to the tradeoff between model complexity and prediction ability, there is no straight forward optimal solution, this tradeoff should be solved.

Although the *PMQ-L* did not find the optimal solution, it did promptly find a good quality solution – a model that efficiently addresses the posed tradeoff. Fig. 7 shows the relative location of the solution – *final model.*

4.2. Case study 2

Sensorless Drive Diagnosis [36], the data set contains 48 numerical features (plus the class label), which are extracted from motor current [37], the motor has good and defective components. This results in 11 different classes with different conditions. The goal of this study is to detect only class one. This data set is highly unbalanced (58509 instances - including 5319 class 1) and it is split as follows: training set (33,409 - including 3100), validation set (12,100 - 1319), and testing set (13, 00–900). Since the data set does not provide specific information about the meaning/name of each feature, they are referred to as feature 1,2, … 48.

The *KNN* learning algorithm is applied with the same number of neighbors, used for the *ReliefF* algorithm. The search space contains 70 pairwise combinations; for *ReliefF* and *KNN*, 7 logarithmically spaced points are defined – $k = \{1, 2, 4, 7, 13, 24, 45\}$ – and for δ, 10 even spaced points – $\delta = \{0.50, 0.55, \ldots, 0.95\}$. Feature relevance is determined by comparing their weights with $\tau = 0.0245$. The *incumbent* model with 3 features (9,11,21) is found with $k = 45$ and $\delta = 0.85$. Then, three *CM* are created (feature 9, features 9,11, features 9,11,21) and the *PMPCD* is used as a *MS* criterion to select the *final model.* According to the criterion, the 3rd model should be selected (*PMPCD* $= 0.9573$), *final model* has an estimated testing *MPCD* $= 0.9857$.

Although the search space of the application of the *KNN* in this data set is subject to combinatorial explosion – $8.7 \times 10^{17}(2^{48} \times 3100)$ – a good quality solution is found after creating 73 models.

4.3. Case study 3

Statlog (Landsat Satellite) [38], the original data set contains 36 features with 7 classes. Only class 1 is considered – class 1 vs all –, the data set is split as follows: training set (4435 - including 1072), validation set (1000–293), and testing set (1000–168).

The *SVM* learning algorithm is applied to the same search space described in *Case Study 2* (70 pairwise combinations). Feature relevance is determined by comparing their weights with $\tau = 0.0672$. The *incumbent* model with 8 features (33,21,25,13,5,14,2,30) is found with $k = 7$ and. Then, 8 CM are created and the *PMPCD* is used as a *MS* criterion to select the *final model.* According to the criterion, the 8th model should be selected (*PMPCD* $=$ 0.8748), *final model* has an estimated testing *MPCD* $= 0.9976$.

4.4. Case study 4

Occupancy Detection [38,39], the data set contains 5 features. To generate an unbalanced data structure, one out of 10 instances labeled as class 1 are included in the data set (index 1, 10, 20, etc.) and the remaining nine eliminated, all 0 class are included. The data set is split as follows: training set (6587 - including 173), validation set (1791 - 98), and testing set (7908 - 205).

The *LR* learning algorithm is applied to the same search space described in *Case Study 2*. Feature relevance is determined by comparing their weights with $\tau = 0.0551$. The *Optimal Classification Threshold with respect to MPCD* algorithm is used to obtain the classification threshold of each *CM* [30]. The *incumbent* model with 2 features (feature 3 - CO_2, feature 1 -Humidity) is found with $k = 1$ and $\delta = 0.50$. Then, 2 CM are created, according to the criterion, the single-feature model should be selected (*PMPCD* $=$ 0.9681), *final model* has an estimated testing *MPCD* $= 0.9879$.

4.5. Discussion

Most of *Big Data* initiatives are subject to feature combinatorial explosion, a situation that gets aggravated by the hyperparameter tuning process e.g., *Case Study 2*. One of the main challenges, is to detect which features actually contain discriminative information, since oftentimes most of them are either irrelevant or redundant. In the four case studies, a good quality solution is found after creating only a few models – with respect to the feature combination space. All these solutions effectively selected only the most relevant features to model the pattern, since each *final model* is virtually separating the testing data (*MPCD* ≈ 1).

5. Comparative analysis

To evaluate the performance of the *PMQ-L* modeling scheme, two

Table 4
Data sets information, positive class count in parenthesis.

Data	Description	Features	Training set	Validation set	Test set	Ratio (overall)
1	UMW	54	18,495 (20)	12,236 (9)	9500 (7)	0.09[b]
2	Statlog (class 1)	36	4435 (1072)	1000 (293)	1000 (168)	23.82[a]
3	Credit Card Fraud	29	200,000 (385)	40,000 (52)	44,807 (55)	0.17[b]
4	Occupancy Detection	5	6587 (173)	1791 (98)	7908 (205)	2.92[a]
5	HTRU2	8	12,000 (1484)	2000 (91)	3898 (64)	9.16[a]

[a] Highly-unbalanced.
[b] Ultra-unbalanced The preprocessing information can be found in Refs. [43].

Table 5
Solutions by data set, comparative analysis 1.

Data set	l_1-regularized *LR*			PMQ-L		
	Features	Hyperparameter (λ)	Testing *MPCD*	Features	Hyperparameters (k,δ)	Testing *MPCD*
1	42	7.168e-07	0.8567	2	12,0.65	0.9956
2	34	4.721e-05	0.9929	8	1,0.95	0.9976
3	21	2.738e-05	0.8268	10	1,0.50	0.8033
4	5	9.839e-06	0.6784	1	1,0.50	0.9879
5	7	5.520e-05	0.8727	4	1,0.90	0.8758

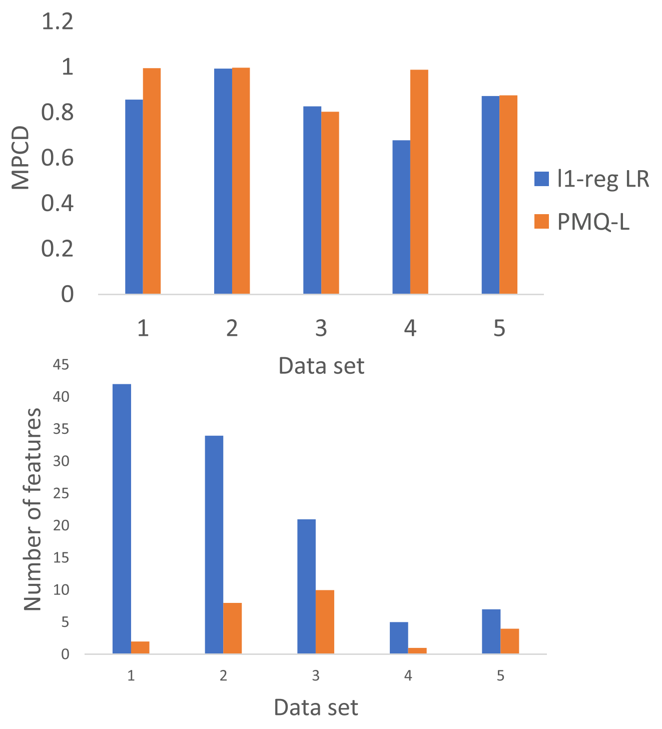

Fig. 8. Comparative analysis, *MPCD* and number of features by data set.

comparative analyses are presented: (1) vs. the l_1-regularized *LR* learning algorithm [19], this algorithm induces parsimony, therefore the goal of this analysis is to evaluate how *PMQ-L* solves the posed tradeoff between complexity and parsimony; (2) vs. the *Random Undersampling Boosting (RUSBoost)* learning algorithm [40], this algorithm is designed specifically to analyze highly/ultra unbalanced data structures, but it does not induce parsimony, therefore prediction analysis is the main goal of this comparative study.

Five highly/ultra-unbalanced data sets are analyzed, Table 4. In addition to the data sets of case studies 1,3,4, two publicly available data sets are also included in this analysis.[2] First, for each data set 70 CM are developed and the *final model* selected using the *PMQ-L* modeling scheme. Then, following the learning approach in Ref. [30], the l_1-regularized *LR* learning algorithm is applied to the same data set to develop 100 CM by varying the regularization values (λ) [41]. Finally, the Akaike information criterion [42] is used to select the *final model*. The solutions of the two learning approaches are evaluated in terms of the number of *CM* developed, the number of features in the *final model* and their generalization ability. Results are presented in Table 5. For reproducibility purposes the hyperparameters values of the *final models* are included.

According to experimental results, four (data sets 1,2,4,5) out of five solutions of the proposed learning scheme outperforms the l_1-regularized *LR*-based solutions, since they have a lesser number of features and exhibit better generalization performance. In the third solution, the tradeoff is solved differently. The *final model* developed by *PMQ-L* exhibits slightly lower generalization ability (0.8033 vs 0.8268), but it includes significantly smaller number of features (10 vs 21). This comparative analysis is graphically presented in Fig. 8.

A second comparative analysis with a widely used learning algorithm in the category of over/under-sampling method is presented. The *RUSBoost* is a combination of random undersampling and *AdaBoost* [44] specifically designed to analyze highly/ultra unbalanced data structures. Random undersampling is applied to the majority class to balance the ratio between minority and majority classes, then *AdaBoost* is applied to the balanced-subset to build a model. For this analysis, the *RUSBoost* is applied to the 4 data sets[3] of the case studies presented in Section 4. With a search space of 10–150 trees, the testing results of each of the *final models* are summarized in Table 6.

In this analysis, highly/ultra unbalanced data structures that exhibit both, linear and non-linear patterns are analyzed. According to empirical results, the *PMQ-L* developed better predictive models than *RUSBoost* in all data sets, Table 6. Since in most of the cases, because of the hyperdimensional feature spaces, the pattern is not known in advance, therefore it is recommended to apply all the learning algorithms that can be handled by *PMQ-L* to find the best one.

[2] Since the data set of case study 2 does not exhibit a linear pattern (classes cannot be separated by a linear classifier), it is not included in this analysis.

[3] Since the *RUSBoost* can handle linear and non-linear patterns the four data sets are analyzed.

Table 6
Solutions by data set, comparative analysis 2.

Case study	*RUSBoost*			PMQ-L		
	Features	Hyperparameter (*trees*, *learning rate*)	Testing *MPCD*	Features	Hyperparameters (k,δ)	Testing *MPCD*
1	54	40,0.1	0.9980	2	2,0.50	0.9995
2	48	140,0.1	0.8838	3	45,0.85	0.9857
3	36	140,0.1	0.8377	8	7,0.95	0.9976
4	5	10,0.1	0.9883	1	1,0.50	0.9885

6. Conclusions

A new *Hybrid Feature Selection and Pattern Recognition* method with the capacity to learn from the original data set was proposed, *PMQ-L*. It is aimed at detecting rare quality events through parsimonious modeling. Although the proposed approach does not guarantee to find the optimal solution (if it exists), it did promptly find a good quality solution. Its unique ability to deal with highly/ultra-unbalanced data structures and diverse learning algorithms to model linear and no-linear patterns was demonstrated in the 4 case studies, which also exhibited its capacity of selecting the driving features of the system.

According to empirical results, the proposed modeling scheme outperformed widely-used modeling approaches based on the l_1-regularized logistic regression and the *Random Undersampling Boosting* learning algorithms in terms of parsimony, generalization ability and the number of candidate models needed to develop a good solution.

Since rare event detection and information extraction are two of the main modern challenges in the application of *ML* across industries, the proposed modeling approach can be generalized to other domains – as supported by the case studies – facing the same challenges.

In this research, hyperparameters (k, δ) optimization was performed with respect to validation *MPCD* only. If it is considered that greater separability between classes is preferred for generalization purposes. Future research along this path, can focus on formulating the model assessment task as a two objective optimization problem. In which separability would be the second fitness attribute to be considered to further discriminate between two or more competing models.

Credit author statement

Carlos Alberto Escobar: Developed the method and led all the sections and analysis. Ruben Morales-Menendez: Helped to run the comparative analyses and contrast the contribution of the method. Daniela Macias Arregoyta: Collaborated with the literature review of the paper.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] Kumar V, Mathur P, Sahni R, Anand M. Two-dimensional multi-release software reliability modeling for fault detection and fault correction processes. Int J Reliab Qual Saf Eng 2016;23(3):164–78.
[2] He Y, Liu F, Cui J, Han X, Zhao Y, Chen D, Zhou Z, A Z. Reliability-oriented design of integrated model of preventive maintenance and quality control policy with time-between-events control chart. Comput Ind Eng 2019;(129):228–38.
[3] Cai B, Zhao Y, Liu H, Xie M. A data-driven fault Diagnosis methodology in three-phase inverters for PMSM drive systems. IEEE Trans Power Electron 2016;32(7):5590–600.
[4] Kang S, Kim E, Shim J, Cho S, Chang W, Kim J. Mining the relationship between production and customer service data for failure analysis of industrial products. Comput Ind Eng 2017;106:137–46.
[5] Abell JA, Chakraborty D, Escobar CA, Im KH, Wegner DM, Wincek MA. Big data driven manufacturing — process-monitoring-for-quality philosophy. ASME J Manuf Sci Eng Data Sci Enhanc Manuf 2017;139(10).
[6] Ribeiro MT, Singh S, Guestrin C. Why should I trust you?: explaining the predictions of any classifier. In: Proc of the 22nd ACM SIGKDD int Conf on knowledge Discovery and data mining; 2016. p. 1135–44.
[7] Gunning D. Explainable artificial intelligence (XAI). *Defense Advanced Research Projects Agency*; 2017.
[8] Shao C, Paynabar K, Kim T, Jin J, Hu S, Spicer J, Wang H, Abell J. Feature selection for manufacturing process monitoring using cross-validation. J Manuf Syst 2013;10.
[9] Wuest T, Weimer D, Irgens C, Thoben K-D. Machine learning in manufacturing: advantages, challenges, and applications. Prod Manuf Res 2016;4(1):23–45.
[10] Yu L, Liu H. Feature selection for high-dimensional data: a fast correlation-based filter solution. In: ICML, vol. 3; 2003. p. 856–63.
[11] Haixiang G, Yijing L, Shang J, Mingyun G, Yuanyue H, Bing G. Learning from class-imbalanced data: review of methods and applications. Expert Syst Appl 2017;73:220–39.
[12] He H, Ma Y. Imbalanced learning: foundations, algorithms, and applications. John Wiley & Sons; 2013.
[13] Shukla S, B.S. R. Online sequential class-specific extreme learning machine for binary imbalanced learning. Neural Network 2019;119(235–248).
[14] Chandrashekar G, Sahin F. A survey on feature selection methods. Comput Electr Eng 2014;40(1):16–28.
[15] Ng A. On feature selection: learning with exponentially many irrevelant features as training examples. In: *Proc of the* 15*th* *int Conf on machine learning.* MIT, Dept. of Electrical Eng and Computer Science; 1998. p. 404–12.
[16] De Silva AM, Leong PH. Feature selection. In: Grammar-based feature Generation for time-series prediction. Springer; 2015. p. 13–24.
[17] Deng H, Runger G. Feature selection via regularized trees,. In: Int J Conf on neural networks; 2012. p. 1–8.
[18] Weston J, Mukherjee S, Chapelle O, Pontil M, Poggio T, Vapnik V. Feature selection for SVMs. In: NIPS, vol. 12; 2000. p. 668–74.
[19] Tibshirani R. Regression shrinkage and selection via the LASSO,. J Roy Stat Soc B 1996:267–88.
[20] Ng A. Feature selection L1 vs L2 regularization and rotational invariance. In: *Proc. Of the* 21*st* *int Conf on machine learning.* ACM; 2004. p. 78.
[21] Wang F, Yang Y, Lv X, Xu J, Li L. Feature selection using feature ranking, correlation analysis and chaotic binary particle swarm optimization. In: 5*th* *int Conf on software Eng and service science*; 2014. p. 305–9.
[22] Kira K, Rendell L. The feature selection problem: traditional methods and a new algorithm. In: AAAI, vol. 2; 1992. p. 129–34.
[23] Robnik-Šikonja M, Kononenko I. Theoretical and empirical analysis of ReliefF and RReliefF. Mach Learn 2003;53(1–2):23–69.
[24] Kononenko I. Estimating attributes: analysis and extensions of RELIEF. In: European Conf on machine learning. Springer; 1994. p. 171–82.
[25] Devore J. Probability and statistics for engineering and the sciences. Cengage Learning; 2015.
[26] Fawcett T. An introduction to ROC analysis. Pattern Recogn Lett 2006;27(8):861–74.
[27] Escobar CA, Morales-Menendez R. Process-Monitoring-for-Quality — a model selection criterion. SME Manuf Lett 2018;15:55–8.
[28] Arlot S, Celisse A. A survey of cross-validation procedures for model selection. Stat Surv 2010;4:40–79.
[29] Friedman J, Hastie T, Tibshirani R. The elements of statistical learning, vol. 1. Berlin: Statistics Springer; 2001.
[30] Escobar CA, Morales-Menendez R. Machine learning techniques for quality control in high conformance manufacturing environment. Adv Mech Eng 2018;10(2):1–12.
[31] The MathWorks Inc. Logspace. [Online]. 2017. Available: www.mathworks.com/help/matlab/ref/logspace.html.
[32] Kuhn M, Johnson K. Applied predictive modeling, vol. 26. Springer; 2013.
[33] Bergstra J, Bengio Y. Random search for hyper-parameter optimization. J Mach Learn Res 2012;13(2):281–305.
[34] Claesen M, De Moor B. "Hyperparameter search in machine learning. In: The XI metaheuristics int conf; 2015.
[35] Murphy K. Machine learning: a probabilistic perspective. MIT press; 2012.
[36] Lichman M. UCI machine learning repository. 2013 [Online]. Available: http://archive.ics.uci.edu/ml.
[37] Paschke F, Bayer C, Bator M, Mönks U, Dicks A, Enge-Rosenblatt O, Lohweg V. Sensorlose zustandsüberwachung an synchronmotoren. In: *Proc* 23*th. Workshop computational intelligence*, vol. 5; 2013. p. 211. Dortmund, Germany.
[38] Dheeru D, Karra Taniskidou E. UCI machine learning repository. 2017 [Online]. Available, http://archive.ics.uci.edu/ml.
[39] Candanedo LM, Feldheim V. Accurate occupancy detection of an office room from light, temperature, humidity and CO_2 measurements using statistical learning models. Energy Build 2016;112:28–39.

[40] Seiffert C, Khoshgoftaar TM, Van Hulse J, Napolitano A. Rusboost: a hybrid approach to alleviating class imbalance. IEEE Trans Syst Man Cybern Syst Hum 2009;40(1):185–97.

[41] T. M. Inc. Lasso. 2011 [Online]. Available: https://www.mathworks.com/help/stats/lasso.html.

[42] Burnham KP, Anderson DR. Model selection and multimodel inference: a practical information-theoretic approach. Springer Science & Business Media; 2003.

[43] Escobar CA, Morales-Menendez R. Process-monitoring-for-quality—a model selection criterion for l1-regularized logistic regression. Procedia Manuf 2019;34: 832–9.

[44] Freund Y, Schapire RE, et al. Experiments with a new boosting algorithm. In: icml, vol. 96. Citeseer; 1996. p. 148–56.