Francisco de Assis Boldt

Classifier Ensemble Feature Selection for Automatic Fault Diagnosis

Vitória - Espírito Santo - Brazil

14 July 2017

Francisco de Assis Boldt

Classifier Ensemble Feature Selection for Automatic Fault Diagnosis

Doctoral Thesis presented according to the regiment of the Programa de Pós-graduação em Informática of the Universidade Federal do Espírito Santo.

Universidade Federal do Espírito Santo

Departamento de Informática

Doutorado em Ciência da Computação

Advisor: Dr. Thomas Walter Rauber

Co-advisor: Dr. Flávio Miguel Varejão

Vitória - Espírito Santo - Brazil

14 July 2017

Dados Internacionais de Catalogação-na-publicação (CIP)
(Biblioteca Setorial Tecnológica,
Universidade Federal do Espírito Santo, ES, Brasil)

Boldt, Francisco de Assis, 1976-
B687c Classifier ensemble feature selection for automatic fault diagnosis / Francisco de Assis Boldt. – 2017.
110 f. : il.

Orientador: Thomas Walter Rauber.
Coorientador: Flávio Miguel Varejão.
Tese (Doutorado em Ciência da Computação) – Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Localização de falhas (Engenharia). 2. Classificadores. 3. Seleção de características (Computação). 4. Seleção de recursos. I. Rauber, Thomas Walter. II. Varejão, Flávio Miguel. III. Universidade Federal do Espírito Santo. Centro Tecnológico. IV. Título.

CDU: 004

Francisco de Assis Boldt

Classifier Ensemble Feature Selection for Automatic Fault Diagnosis

Doctoral Thesis presented according to the regiment of the Programa de Pós-graduação em Informática of the Universidade Federal do Espírito Santo.

Work approved. Vitória - Espírito Santo - Brazil, 14 July 2017:

Prof. Dr. Thomas Walter Rauber Advisor	**Prof. Dr. Flávio Miguel Varejão** Co-advisor
Prof. Dr. Evandro Ottoni Teatini Salles Guest	**Prof. Dr. André Carlos Ponce de Leon Ferreira de Carvalho** Guest
Prof. Dr. Thiago Oliveira dos Santos Guest	**Prof^a. Dr^a. Aura Conci** Guest

Vitória - Espírito Santo - Brazil
14 July 2017

Abstract

An efficient ensemble feature selection scheme applied for fault diagnosis is proposed, based on three hypothesis:

a. A fault diagnosis system does not need to be restricted to a single feature extraction model, on the contrary, it should use as many feature models as possible, since the extracted features are potentially discriminative and the feature pooling is subsequently reduced with feature selection;
b. The feature selection process can be accelerated, without loss of classification performance, combining feature selection methods, in a way that faster and weaker methods reduce the number of potentially non-discriminative features, sending to slower and stronger methods a filtered smaller feature set;
c. The optimal feature set for a multi-class problem might be different for each pair of classes. Therefore, the feature selection should be done using an one versus one scheme, even when multi-class classifiers are used. However, since the number of classifiers grows exponentially to the number of the classes, expensive techniques like Error-Correcting Output Codes (ECOC) might have a prohibitive computational cost for large datasets. Thus, a fast one versus one approach must be used to alleviate such a computational demand.

These three hypothesis are corroborated by experiments.

The main hypothesis of this work is that using these three approaches together is possible to improve significantly the classification performance of a classifier to identify conditions in industrial processes. Experiments have shown such an improvement for the 1-NN classifier in industrial processes used as case study.

Keywords: Classifier Ensemble, Feature Selection, Automatic Fault Diagnosis.

Contents

Acronyms

ACO Ant Colony Optimization. 39

ANN Artificial Neural Network. 21, 23, 25

AUC-ROC area under the ROC curve. 34

Bagging **b**ootstrap **agg**regat**ing**. 40

CF crest factor. 45

CFS Cascade Feature Selection. 75

CWRU Case Western Reserve University. 43, 99

ECOC Error-Correcting Output Codes. 16

ELM Extreme Learning Machine. 21, 28

ESP Electric Submersible Pump. 55, 99

FC frequency center. 46

FER Facial Expression Recognition. 42

FFT Fast Fourier Transform. 46

G-mean Geometric Mean. 33

GA Genetic Algorithm. 39, 41

GEFS Genetic Ensemble Feature Selection. 41

IF impulse factor. 45

k-NN k-Nearest Neighbor. 21, 22

KF kurtosis factor. 45

KV kurtosis value. 45

LS Least Square. 25

MF margin factor. 45

MLE Maximum Likelihood Estimator. 35

MLP Multi Layer Perceptron. 25

mRMR minimal Redundancy Maximal Relevance. 38

MVN Multivariate Normal. 35

PCA Principal Component Analysis. 13, 67

PPV peak-peak value. 45

PSO Particle Swarm Optimization. 39

RAM Random-Access Memory. 23

RMS root mean square. 45

RMSF RMS frequency. 46

ROC Receiver Operating Characteristic. 34

RVF root variance frequency. 46

SBS Sequential Backward Selection. 39, 41, 65

SF shape factor. 45

SFBS Sequential Floating Backward Search. 39, 65

SFFS Sequential Floating Forward Search. 39, 65

SFS Sequential Forward Selection. 37, 39, 41, 65

SRA square root of the amplitude. 45

SRKNN Sequential Random K-Nearest Neighbor. 42

SV skewness value. 45

SVM Support Vector Machine. 14, 21, 26, 41

TE Tennessee Eastman Chemical Process. 51, 99

List of Figures

List of Tables

1 Introduction

Automatic fault diagnosis of complex machinery has economical and security related advantages. Identifying a fault in its initial stage allows the early replacement of damaged parts. This type of predictive maintenance is better than the preventive counterpart, which replaces parts that are not necessarily defective. Pattern recognition techniques have been used for automatic fault diagnosis of industrial processes (BOLDT et al., 2017; RAUBER et al., 2017; BOLDT; RAUBER; VAREJÃO, 2017), to develop model free diagnosis systems. These kind of systems need to extract relevant data from the problem domain in order to train classification algorithms. Several model free fault diagnostic systems proposed in the scientific literature can be split in three main modules, as presented in figure 1. Despite the inexistence of a hard margin separating the three modules, each one has a specific area which it is better related. The data acquisition module is related with the engineering area. It provides the problems that have to be solved and gives some tools that can be used by human specialists or intelligent systems to construct automatic fault diagnosis systems. The machine learning module is related with computer science area. The contributions of this proposal are strongly related with this module, e.g. Feature Extraction, Feature Selection and Classifier Ensembles. However, there are contributions to the other two modules in a minor form, e.g. multiple paired nested cross-validation. The assessment module is related with the statistics area. It is responsible, among other duties, to evaluate and compare different diagnosis systems.

Each sub-module in figure 1 has some relation with all three main modules. Nevertheless, some of them has stronger relation with one instead the others. For instance, classifiers are frequently based on statistical foundations, but they are better related with the machine learning module than the assessment one. Another example is the classification based on threshold. This kind of classification applies only few knowledge from the machine learning area. On the other hand, it is easy to find relevant papers that uses a single threshold over the acquired data (YIN et al., 2012; YIN; LUO; DING, 2014). Lines in figure 1 represent possible diagnosis systems in the modular arrangement. An example of diagnosis system that uses a single threshold instead a more complex classifier presented by YIN; LUO; DING (2014). The dotted light gray line represents this kind of process. A comparison with feature extraction on the feature level using Principal Component Analysis (PCA), represented by the dotted dark gray line in figure 1, is presented in RAUBER; BOLDT; VAREJÃO (2015). In BOLDT et al. (2015) and BOLDT; RAUBER; VAREJÃO (2014b) automatic diagnosis systems that follow the dashed light gray line in figure 1 are presented. The system construction starts with the acquisition of the data by accelerometers. From the acquired vibratory data, features are extracted using several

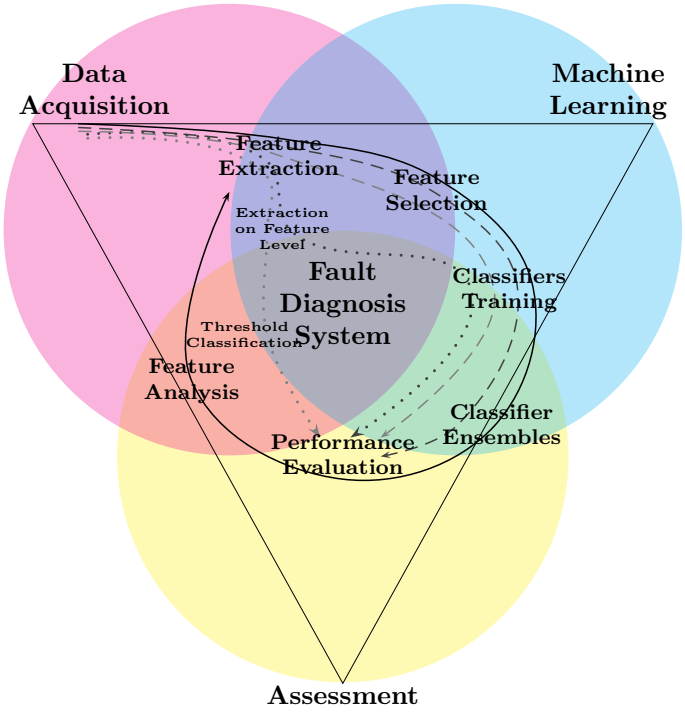


Figure 1 – Modules to develop and assess Fault Diagnosis Systems.

extraction models to create a dataset. Some features are selected before training the final classifier. The evaluation performance compares different feature extraction models and feature selection approaches. This approach added by a classifier ensemble feature selection is represented by the dashed dark gray line and was presented in BOLDT et al. (2017). The types of automatic diagnosis system do not need to be restricted to these four examples and also do not need to end in the performance evaluation. The continuous black line gives an example of a hypothetical iterative system that uses the evaluation to improve the extraction methods and, consequently, whole system. Such an interactive system would be related with on-line diagnosis systems, when the systems continue to learn while they are been used. This kind of systems are out of the scope of this work.

1.1 Motivation

A model-free diagnosis system needs a large amount of examples to be constructed (WANDEKOKEM et al., 2011). To evaluate and compare these systems with strong statistical methods is sometimes unfeasible (BOLDT et al., 2017). Statistical tests that require a large number of repetitions (BOUCKAERT, 2004) cannot be performed in a reasonable time when computationally expensive methods are used to create model-free diagnosis systems for datasets with large amount of features and samples (BOLDT et al., 2015). One example of expensive method is given in WANDEKOKEN et al. (2011), where Support Vector Machine (SVM) classifier ensembles are created by the variation of the SVM hyper-parameters. Two ensemble methods are compared with a large amount of

data. However, the comparison did not use a strong statistical method and it was not bias aware. Therefore, despite the interesting contribution of WANDEKOKEN et al. (2011), there is no statistical evidence that the presented method is better or worse than others to be applied in practical diagnosis systems. On the other hand, OLIVEIRA-SANTOS et al. (2016) presents a statistically strong comparison among different classifiers to diagnose faults in submersible motor pumps. Such a comparison would be unfeasible for expensive classifier ensemble feature selection methods, because this type of statistical test demands a large number of repeated experiments.

One may argue that the diagnosis systems do not need to be trained in a fast manner, because the training phase is not frequently done. However, without statistically strong tests, there is not guarantee of which methods are the most appropriated for each context. Therefore, complex pattern recognition systems that uses expensive methods like ensembles are desirable, but they have to be fast enough to be tested in a way that provides some confidence to choose one method instead another.

1.2 Contributions

This proposal presents a classifier ensemble feature selection method applied to fault diagnosis that is fast enough to be compared by statically significant methods as those presented in OLIVEIRA-SANTOS et al. (2016) and CORANI et al. (2016). This main contribution can be divided in three specific ones.

Heterogeneous Feature Models This contribution is based on the hypothesis that automatic fault diagnosis systems do not need to be restricted to a single feature extraction model. Using heterogeneous feature extraction models to generate a pool of features, subsequently reduced by a feature selection technique can improve the system classification performance. The paper that presents this hipothesis (RAUBER; BOLDT; VAREJÃO, 2015) have received many citations and its principle have been used by other researchers of the fault diagnosis area (WEI; CHOW; CHAN, 2015; BROETTO; VAREJÃO, 2016).

Cascade Feature Selection Feature selection methods are essentially combinatorial optimization problems. Thus, the brute force solution may be unfeasible for large datasets. Commonly, the system designers have to cope with a trade off between classification performance and speed of training. Therefore, a cascade scheme is proposed to accelerate the feature selection process without loss of performance (BOLDT; RAUBER; VAREJÃO, 2017). Inspired by the cascade classifier method (VIOLA; JONES, 2001) used to identify face in images, fast and weak feature selection methods are placed on the top of the cascade while strong and slow feature selection methods, placed on the bottom of the cascade, have to cope with a reduced

amount of features. Experiments have shown that the cascade schema can improve the final system classification performance at the same time that speed up the feature selection process.

One vs. One Ensemble Feature Selection Based on the hypothesis that different group of faults might require different set of features to maximize their identification rate, an one versus one schema, dividing the dataset classes by pairs to select the features may improve the system classification performance. However, techniques like Error-Correcting Output Codes (ECOC) (DIETTERICH; BAKIRI, 1995) might spend a prohibitive amount of time for large datasets with expensive feature selection methods. Therefore, an efficient method to combine the conditions using an one versus one approach is proposed (BOLDT et al., 2017). Experiments have shown classification performance improvement of the one versus one feature selection approach over the multi-class feature selection, for the same classifier architecture.

The main proposal of this work applies these three contributions together to construct efficient and cost effective classifier ensembles based on feature selection for fault diagnosis. All proposals in this work are focused in the pattern recognition area and not restricted to industrial processes. All of them can be applied for general purposes. The industrial processes presented as case study were used due their scientific importance and practical applicability.

Beyond these theoretical contributions, this work also produced two software frameworks. The first, dedicated to extract statistical features from vibratory signals, implemented in Matlab using object oriented paradigm. It provides "plug & play" interfaces that facilitate to adapt new serial data bases and new feature models. Some feature models are already implemented in the framework, i.e. statistical features from time and frequency domain, wavelet packet analysis, complex envelope analysis, empirical decomposition mode based features. The second, is a classification framework compatible with Matlab and Octave. It has modules with classifiers, performance metrics and validation methods, usually seen in others classifier frameworks. However, this framework allows, in an easy way, to implement new feature selection and ensemble methods. Adaptation of pre-existent classifiers is also possible. The most important parts of this framework are the bias aware capability achieved by nested validations and the possibility of paired experiments, achieved with the identical division of training and test datasets for certain random seed. This kind of evaluation is not frequently present in the most known classification frameworks. The experiments presented in chapter 5 used the classification framework and the experiments presented in chapters 4 and 6 used both frameworks.

1.3 Publications

All contributions that compose this work have been published. Following, the published papers with the Capes Qualis in between parenthesis. For conferences, there are two Capes Qualis. The first is the Capes Qualis published in 2012, that was the valid qualis at the papers submition, and the second is the current valid qualis.

1. BOLDT, F. de A.; RAUBER, T. W.; VAREJÃO, F. M. Cascade feature selection and elm for automatic fault diagnosis of the tennessee eastman process. *Neurocomputing*, v. 239, p. 238 – 248, 2017. ISSN 0925-2312. (A1)

2. RAUBER, T. W.; OLIVEIRA-SANTOS, T.; BOLDT, F. de A.; RODRIGUES, A.; VAREJÃO, F. M.; RIBEIRO, M. P. Kernel and random extreme learning machine applied to submersible motor pump fault diagnosis. In: IEEE. *Neural Networks (IJCNN), 2017 International Joint Conference on*. [S.l.], 2017. (A2/A1)

3. BOLDT, F. de A.; RAUBER, T. W.; OLIVEIRA-SANTOS, T.; RODRIGUES, A.; VAREJÃO, F. M.; RIBEIRO, M. P. Binary feature selection classifier ensemble for fault diagnosis of submersible motor pump. In: *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)*. [S.l.: s.n.], 2017. (B1/B3)

4. RAUBER, T. W.; BOLDT, F. de A.; VAREJÃO, F. M. Heterogeneous feature models and feature selection applied to bearing fault diagnosis. *Industrial Electronics, IEEE Transactions on*, IEEE, v. 62, n. 1, p. 637–646, 2015. (A1)

5. BOLDT, F. de A.; RAUBER, T. W.; VAREJÃO, F. M. Single sequence fast feature selection for high-dimensional data. In: IEEE. *Tools with Artificial Intelligence (ICTAI), 2015 IEEE 27th International Conference on*. [S.l.], 2015. p. 697–704. (A2/B1)

6. BOLDT, F. de A.; RAUBER, T. W.; VAREJÃO, F. M.; RIBEIRO, M. P. Fast feature selection using hybrid ranking and wrapper approach for automatic fault diagnosis of motorpumps based on vibration signals. In: IEEE. *Industrial Informatics (INDIN), 2015 IEEE 13th International Conference on*. [S.l.], 2015. p. 127–132. (B3/B1)

7. BOLDT, F. de A.; RAUBER, T. W.; VAREJÃO, F. M.; RIBEIRO, M. P. Performance analysis of extreme learning machine for automatic diagnosis of electrical submersible pump conditions. In: IEEE. *Industrial Informatics (INDIN), 2014 12th IEEE International Conference on*. [S.l.], 2014. p. 67–72. (B3/B1)

8. BOLDT, F. de A.; RAUBER, T. W.; VAREJÃO, F. M. Evaluation of the extreme learning machine for automatic fault diagnosis of the tennessee eastman chemical pro-

cess. In: IEEE. *Industrial Electronics Society, IECON 2014-40th Annual Conference of the IEEE*. [S.l.], 2014. p. 2551–2557. (B1/B2)

9. BOLDT, F. de A.; RAUBER, T. W.; VAREJÃO, F. M. A fast feature selection algorithm applied to automatic faults diagnosis of rotating machinery. *Journal of Applied Computing Research*, v. 3, n. 2, p. 78–86, 2014. (C)

10. RAUBER, T. W.; BOLDT, F. de A.; VAREJÃO, F. M.; RIBEIRO, M. P. Computational intelligence for automatic diagnosis of submersible motor pump conditions in offshore oil exploration. In: *Electronics, Circuits and Systems (ICECS), 2013 20th IEEE International Conference on*. [S.l.: s.n.], 2013. (B1/B2)

11. RAUBER, T. W.; BOLDT, F. de A.; VAREJÃO, F. M.; RIBEIRO, M. P. Feature models and condition visualization for rotating machinery fault diagnosis. In: IEEE. *Electronics, Circuits, and Systems (ICECS), 2013 IEEE 20th International Conference on*. [S.l.], 2013. p. 265–268. (B1/B2)

12. BOLDT, F. de A.; RAUBER, T. W.; VAREJÃO, F. M. Feature extraction and selection for automatic fault diagnosis of rotating machinery. In: *X Encontro Nacional de Inteligencia Artificial e Computacional (ENIAC).-Fortaleza, Ceará*. [S.l.: s.n.], 2013. p. 213–220. (B4/B3)

Part I

Literature Review

2 Pattern Recognition

Pattern recognition is widely applied in automatic fault diagnosis. The most common approach is the supervised learning (DUDA; HART; STORK, 2001). In the supervised learning approach a set of examples is presented to an algorithm that is able to identify patterns and predict the label of new examples not previously presented. The supervised learning is divided in two categories. When the labels are continuous the learning is known as regression and when the labels are discrete the learning is known as classification. The algorithms used for classification are called classifiers. As the focus of this work is the classification of failures in industrial processes, the next section presents some classifiers architectures, i.e. k-Nearest Neighbor (k-NN), Artificial Neural Network (ANN), Support Vector Machine (SVM) and Extreme Learning Machine (ELM). Section 2.2 presents approaches to evaluate classification methods including validation methods, metrics and statistical tests. Three common approaches for feature selection are shown in section 2.3, i.e. Ranking Feature Selection, Sequential Feature Selection and the use of Heuristics and Metaheuristics to select features. Finally, in section 2.4 a short explanation about classifier ensembles is presented. The most known ensemble methods are presented, i.e. Bagging, Adaboost and Random Forest, as well as the ensemble feature selection.

2.1 Classifiers

In the context of automatic fault diagnosis for industrial processes, an example that uses a classifier could be a system that learns how to identify conditions through examples composed by faulty and non-faulty samples. This system has to be able to identify the condition of new samples that were not presented in the training phase. Figure 2 presents an example of classification process.

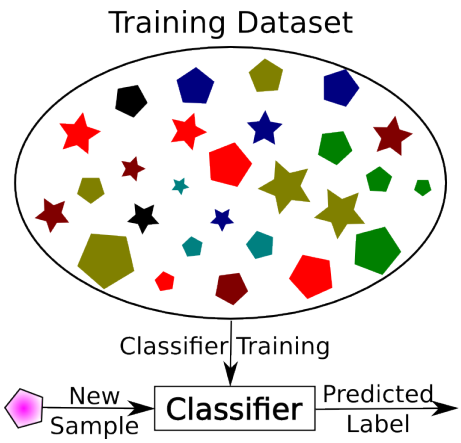


Figure 2 – Classification process.

In figure 2, a set of stars and pentagons are used as training dataset. During training process, the dataset is sent to the classifier in a form of features. These features might be relevant or irrelevant for the correct classification. For instance, the number of angles of each form seems to be relevant to differentiate stars from pentagons. On the other hand, the shapes color does not seem to be relevant for the dataset presented. If all stars had the same color and no pentagon has this specific color, the shapes color could be a good feature to discriminate the samples, but this is not the case of figure 2. Therefore, sometimes it is necessary to select the most important features automatically. The process of discovering relevant features automatically is known as feature selection. Feature selection is discussed in section 2.3. The following subsections explain some classifiers architectures.

2.1.1 K-Nearest Neighbor

The k-Nearest Neighbor (k-NN) Algorithm (COVER; HART, 1967) classifies a new pattern according to the majority vote of its closest neighbors, using, for instance, the Euclidean distance. The benefit of this architecture is its simplicity and its theoretical properties, with respect to the error bound. These properties usually allow this classifier to present a good trade-off between computational cost and classification performance. Figure 3 shows an example of how k-NN classifies a new sample. Considering the training dataset composed by blue squares and red circles in a feature space of two dimensions, a new sample, represented by a green triangle, is presented to a k-NN classifier. A 3-NN classifier would classify the new sample as a red circle, because among the three closest samples to the green triangle in the training dataset two are red circles, as shown by the dotted red line circle. On the other hand, a 5-NN classifier would classify the new sample as a blue square, because among the five closest samples to the green triangle in the training dataset three are blue squares, as shown by the continuous blue line circle.

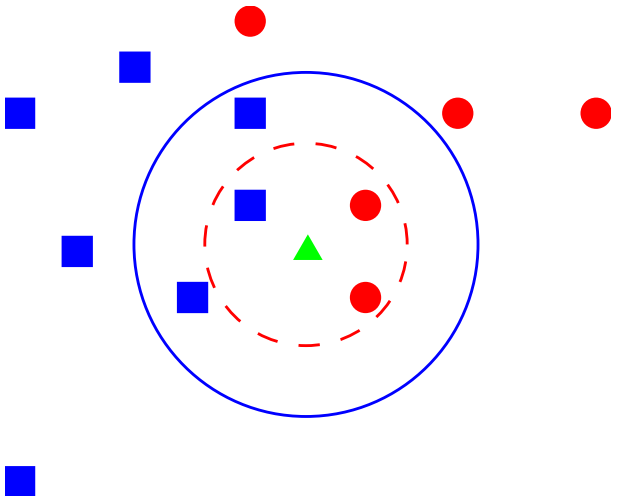


Figure 3 – k-NN Example. A 3-NN classifier would classify the green triangle as a red circle and a 5-NN classifier would classify it as a blue square.

Despite its simplicity, the k-NN algorithm has been widely used in the fault diagnosis context presenting good results, commonly compatibles or superior than more sophisticated algorithms as Support Vector Machine (SVM), Adaboost and Random Forests (OLIVEIRA-SANTOS et al., 2016; RAUBER; BOLDT; VAREJÃO, 2015). This algorithm almost does not take any time in its training phase, since its learning consists in keep the raw training dataset. Using efficient matrix operations present in several linear algebra libraries and softwares like Matlab and Octave, it is possible to implement a k-NN prediction algorithm that answers its prediction relatively fast. That is why k-NN is so attractive for classification. On the other hand, the fact of k-NN has all its learning in the samples makes it to be a prohibitive algorithm for large datasets that cannot be kept in the computers Random-Access Memory (RAM) at once. This would slow down its speed performance drastically. Also, there are situations which the k-NN classifier cannot reach performances higher than those reached by other classifiers, for instance, Artificial Neural Networks.

2.1.2 Artificial Neural Networks: Perceptron, Adaline and Back-propagation

One of the early tries to simulate the human brain was published by MCCULLOCH; PITTS (1943), that proposed a logical calculus of the ideas immanent in nervous activity. This work promoted an innovative research area that resulted in the Perceptron (ROSENBLATT, 1958) and the Adaline (WIDROW; HOFF, 1960). Both of these architectures are single-layer Artificial Neural Network (ANN) models, have a linear decision boundary, use a threshold function and have inside some kind of adaptive linear combiner (WIDROW; LEHR, 2002), as shown in figure 4. The adaptive linear combiner output is a linear combination of its inputs. Its receives at time k an input pattern vector $\mathbf{x}_k = [x_0, x_1, x_2, \cdots, x_n]^T$ and a desired response d_k. The input vector components are weighted by a set of coefficients, the weight vector $\mathbf{w}_k = [w_0, w_1, w_2, \cdots, w_n]$, that is initiated randomly. The linear output is the inner product $s_k = \mathbf{x}_k^T \mathbf{w}_k$. The training process consists in present input patterns with they respective responses to the linear combiner and update the weight vector in each iteration. The update of the weight vector can be done in two different learning modes. In the online learning mode, each iteration k corresponds to a pattern presented. On the other hand, the batch learning mode performs the weight vector update after observing the whole training dataset and each k represents an algorithm iteration over the whole dataset.

Both architectures, Perceptron and Adaline, keep their learning in the weight vector $\mathbf{w}$. In the context of binary classification, the weight vector is perpendicular to the hyperplane that separates the two classes in the features space (DUDA; HART; STORK, 2001). The main difference between the Perceptron and the Adaline is the training rule, as shown in figure 5.

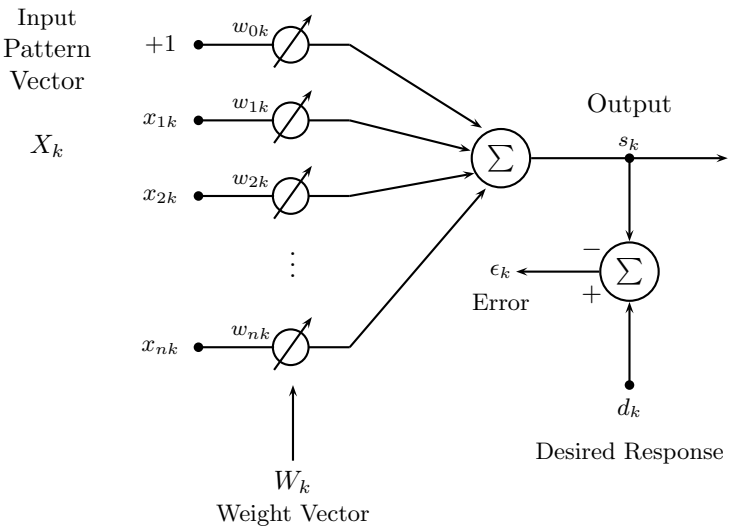
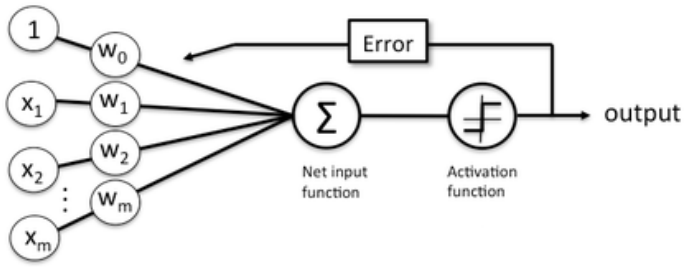


Figure 4 – Adaptive linear combiner (WIDROW; LEHR, 2002).

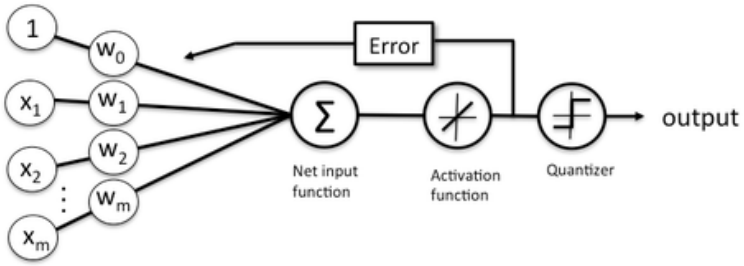
In the Perceptron rule, the linear output is sent to an activation function that converts it into the predicted label. A possible activation function is the step function (EBERHART; SHI, 2011), as equation 2.1.

$$f(x) = \begin{cases} \beta & \text{if } x \geq \theta \\ -\sigma & \text{if } x < \theta \end{cases} \tag{2.1}$$

The differences between the predicted labels and the actual labels of each pattern is used to update the weight vector, as shown in figure 5a. Then, the weight vector $\mathbf{w}$ is updated using the equation 2.2, where η is the constant learning rate and e is the error, calculated as the difference between the predicted label and the actual label, that can be described



(a) Perceptron rule.



(b) Adaline rule.

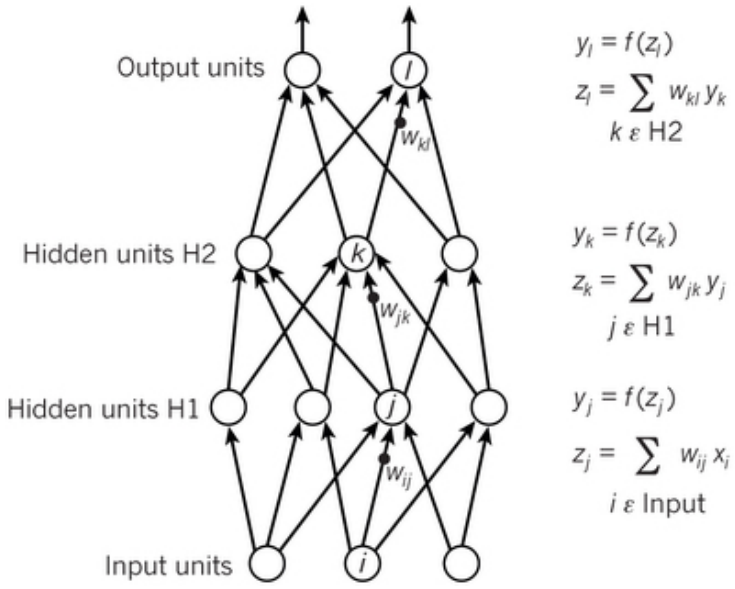
Figure 5 – Difference between Perceptron and Adaline training (RASCHKA, 2015).

as $e_k = d_k - y_k$, where d_k is the actual answer and y_k is the predicted answer for a given pattern (BRAGA; CARVALHO; LUDERMIR, 2007). In other words, $d_k = f(s_k)$, where $f(\cdot)$ is an activation function, e.g. equation 2.1. Considering the binary problem with labels 1 and -1, the error has 1, 0 and -1 as possible values.

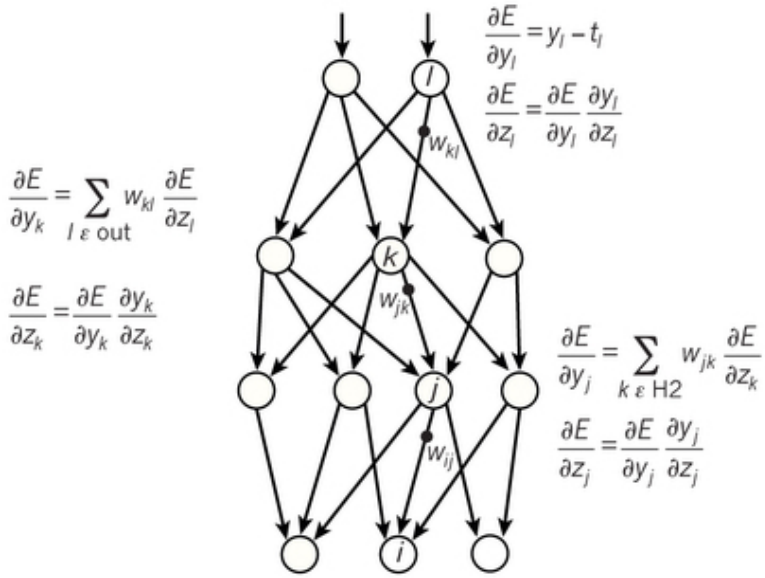
$$\mathbf{w}_{k+1} = \mathbf{w}_k + \eta e \mathbf{x}_k \tag{2.2}$$

Adaline does not use predicted labels to calculate the error and then update the weight vector. Instead, it uses uses continuous predicted values to learn the model coefficients, since its activation function is a linear function ($f(x) = \alpha x$), as shown in figure 5b. Commonly, the value of α is set to 1, that is equivalent to remove the activation function completely (EBERHART; SHI, 2011). However, the prediction is a discrete value, since the continuous value obtained by the inner product $s_k = \mathbf{x}_k^T \mathbf{w}_k$ is sent to a quantizer, that can be a step function like equation 2.1. This update process is known as Widrow-Hoff delta rule (RUMELHART; HINTON; WILLIAMS, 1985) or Least Square (LS) algorithm (WIDROW; HOFF, 1960). The minimum least square solution found by the LS algorithm can also be obtained by means of the Moore-Penrose pseudoinverse $\mathbf{X}^\dagger = (\mathbf{X}^\intercal \mathbf{X})^{-1} \mathbf{X}^\intercal$ (SMOLENSKY; MOZER; RUMELHART, 2013) as $\mathbf{w} = \mathbf{X}^\dagger \mathbf{d}$, where $\mathbf{X}$ is the matrix with the patterns, $\mathbf{d}$ is a vector with the desired labels of the patterns and $\mathbf{w}$ is the weight vector. This is the algorithm used by the Extreme Learning Machine, a single-hidden layer neural network architecture discussed in section 2.1.4.

After MINSKY; PAPERT (1969) highlighting the limitations of the linear separability, many researchers abandoned the connectionism approach (BRAGA; CARVALHO; LUDERMIR, 2007). It was known that multi-layer neural networks do not have the linear separability restriction. However, it was not known an algorithm to training such an architecture. An algorithm to training the multi-layer architecture was proposed by WERBOS (1974). But, it had not much attention from the researchers at that time. When WILLIAMS; HINTON (1986) was published, it brought back the back-propagation algorithm proposal. Then, the neural networks became popular among researchers again. Since the publication of WILLIAMS; HINTON (1986) until nowdays, the back-propagation algorithm is one of the most popular methods for training multi-layer networks based on gradient descent error (DUDA; HART; STORK, 2001). It is based on the delta rule proposed by WIDROW; HOFF (1960), thus it is also called generalized delta rule (BRAGA; CARVALHO; LUDERMIR, 2007). It has two phases, as shown in figure 6. In the feed-forward phase (figure 6a), the pattern is presented to the network that calculates the outputs of each hidden layer up to the output layer. Then, the error is computed and the back-propagation phase (figure 6b) adjust the weights of each layer, in the reverse order of the feed-forward phase. When a multi-layer ANN is trained with the back-propagation algorithm based on the perceptron rule, it might be also called Multi Layer Perceptron (MLP).



(a) Feed-forward information.



(b) Back-propagation error correction.

Figure 6 – Two phases of back-propagation (LECUN; BENGIO; HINTON, 2015).

2.1.3 Support Vector Machine

The Support Vector Machine (SVM) (BURGES, 1998) training algorithm creates a maximum-margin separation hyperplane between two classes, as shown in figure 7. The maximum-margin separation hyperplan algorithm was initially constructed to be a linear classifier (VAPNIK; KOTZ, 1982). However, in order to enhance the linear separation in the original Euclidean space, the SVM maps the input vectors into a high-dimensional feature space through some nonlinear mapping (VAPNIK, 1999), using a kernel function, as shown in figure 8. To classify more than two classes, one may use an one-against-all approach.

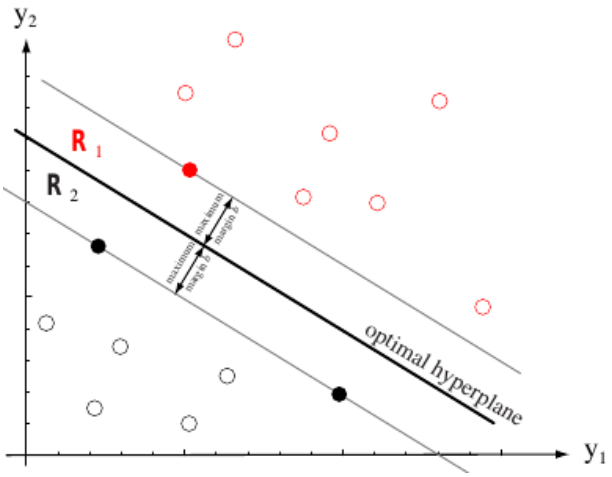


Figure 7 – The Support Vector Machine finds the hyperplane with the maximum distance from the nearest training patterns (DUDA; HART; STORK, 2001).

Currently, the SVM is one of the most used classifiers in fault diagnosis (WANDEKOKEN et al., 2011). It presents excellent classification performance. However, despite its training process be relatively fast, its classification performance is highly dependent of its kernel hyperparameters. To find the optimal value to these hyperparameters is necessary a tuning phase, that demands a relatively large amount of time. Moreover, experiments not properly designed with nested validation (section 2.2.1) might present over-optimist results caused by biased hyperparameters set. As bias aware experiments usually demands highly computational power, the SVM architecture might be unpractical when it is used with feature selection and ensemble methods for large datasets. If robust statistical tests are envisioned, the performance issue becomes even more sensible.

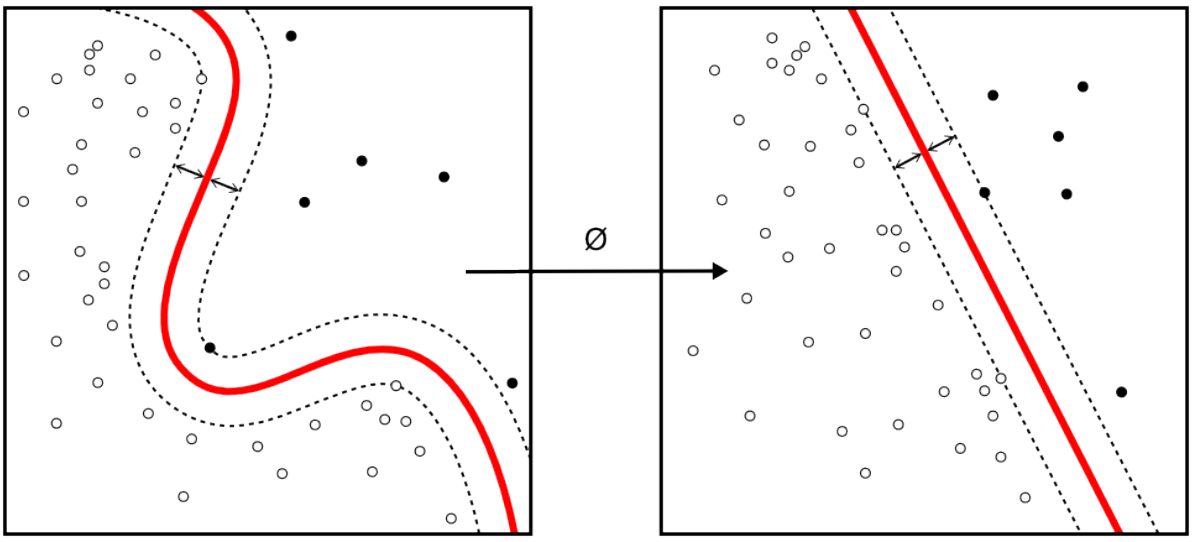


Figure 8 – SVMs using the kernel trick are able to compute non-linearly separable functions into a higher dimension linearly separable function (Support vector machine, 2017).

2.1.4 Extreme Learning Machine

The Extreme Learning Machine (ELM) is a single hidden layer feedforward neural network, with a linear activation function at the output. Unlike multilayer perceptrons trained by gradient descent, for instance error backpropagation, ELM does not adjust the input-to-hidden weights, just choosing them randomly, as well as the bias values. The hidden-to-output weights are deterministically calculated, commonly by multiplying the Moore-Penrose pseudoinverse of the values of the hidden layer units of all patterns with their respective targets. The price of just attributing random weights in the input-to-hidden layer is the necessity of usually having to provide a very large number of hidden units, in order to capture the randomly generated discriminative features. In TAPSON; SCHAIK (2013) the methodology was summarized as follows:

1. Connect the input layer to a much higher number of units in the hidden layer and choose the weights randomly.

2. Calculate the output weights connecting the hidden layer to the output neurons using the pseudoinverse of the product of the hidden layer activations and the target outputs.

The ELM method has been widely and rapidly adopted due to characteristics such as a single forward computational step, a least-squares optimal, non-parametrized solution and computational equivalency to the Support Vector Machine (TAPSON; SCHAIK, 2013). The ELM implementation used in this work is an adaptation of that implemented by the ELM authors in Matlab (HUANG; ZHU; SIEW, 2006). The number of hidden layer neurons was heuristically set to ten times the number of input neurons as suggested by TAPSON; SCHAIK (2013).

Figure 9 shows the mapping structure of the ELM. Consider as input to the net a d-dimensional pattern $\mathbf{x}$ from an input domain which is usually the Euclidean vector space $\mathbb{R}^d$. The net input to unit i in the hidden layer is

$$n_i = \sum_{l=1}^{d} w_{i,l} x_l + b_i = \mathbf{w}_i \cdot \mathbf{x} + b_i, \tag{2.3}$$

where $w_{i,l}$ is the weight between the l-th input and the i-th hidden unit, and $b_i \in \mathbb{R}$ is the associated bias. Passing through an activation function f, the output of unit i in the hidden layer becomes

$$h_i = f(n_i), \tag{2.4}$$

and all L components can be grouped into a vector $\mathbf{h} = [h_1 \cdots h_i \cdots h_L]^{\mathsf{T}}$. The activation function $f : \mathbb{R} \to \mathbb{R}$ is usually the logistic sigmoid function $f(n) = 1/(1 + \exp(-n))$, hyperbolic tangent sigmoid function $f(n) = \tanh n$ or the radial basis function $f(n) =$

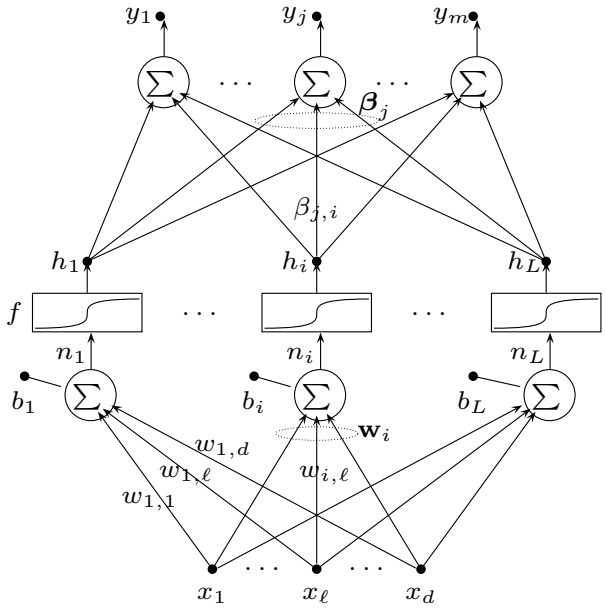


Figure 9 – Architecture of the Extreme Learning Machine.

$\exp(-n^2)$. The identity function $f(n) = n$ cannot be used as activation in the hidden layer, since this would cause a global linear mapping made by the architecture.

A distinct characteristic of the ELM is that the $(d \times L)$ weights $w_{i,l}$ and L biases b_i are randomly initialized and then frozen. Hence, subsequent learning in the hidden-to-output layer has to consider exclusively the new patterns $\mathbf{h}$ of the hidden layer. The number L of hidden units must be much higher then the dimension d of the original feature vector, i.e. $L \gg d$. This is necessary to augment the chances to produce useful new features by the random mapping of the hidden layer. As a rule of thumb one can consider $L = 10 \cdot d$. The output layer of the ELM is composed of $j = 1, \ldots, m$ units y_j which are simply linear combinations of the hidden units h_i. Hence the final output is calculated as

$$y_j = \sum_{i=1}^{L} \beta_{j,i} h_i = \boldsymbol{\beta}_j \cdot \mathbf{h}. \tag{2.5}$$

Again the components can be grouped into the output vector $\mathbf{y} = [y_1 \cdots y_j \cdots y_m]^\mathsf{T}$.

2.2 Performance Evaluation

The performance evaluation is more related with the assessment module in figure 1. It includes validation methods, classification performance metrics and statistical significance tests. The performance evaluation is usually compounded by validation method and a performance metric. The validation method refers to how the data available is used to training and test a classification method. The performance metric defines how to measure the classification performance of a method. Several metrics, among those most used, are based on the confusion matrix generated by some validation method. The confusion matrix

contains the number of samples that were correct and incorrect classified by some method. Table 1 shows an example of a confusion matrix for binary classification.

Table 1 – Confusion matrix for binary classification and the corresponding array representation used in SOKOLOVA; LAPALME (2009).

Data Class	Classified as Positive	Classified as Negative
Actually Positive	true positive (tp)	false negative (fn)
Actually Negative	false positive (fp)	true negative (tn)

For multi-class problems, the numbers of columns and rows of the confusion matrix is equal to the number of the classes, as table 2 shows for a three classes hypothetical multi-class problem. It is important to remember that in multi-class problems each sample has one class associate to it, unlike multi-label problems that allow one sample to have more than one class simultaneously. This work does not study multi-label problems. Moreover, this kind of problem has its own performance measures. Tables 1 and 2 are used to calculate the metrics presented in section 2.2.2.

Table 2 – Confusion matrix for a multi-class problem with three classes.

Data Class	Classified as A	Classified as B	Classified as C
Actually A	true A	false B	false C
Actually B	false A	true B	false C
Actually C	false A	false B	true C

2.2.1 Validation

Publications that perform experiments using datasets that have no explicit separation of training and test may use cross-validation to evaluate and compare different techniques (PARK; KIM, 2015). However, a single cross-validation has high variability due to its randomness. Different fold separations may result in significantly different values, favoring one method over other. Consequently, the replicability of this validation method is low. Replicability of an experiment is a measure of how well the outcome of an experiment can be reproduced (BOUCKAERT, 2004). A stronger approach, also very common in fault detection, is the repeated cross-validation, e.g. ten rounds of 10-folds cross-validations (RAUBER; BOLDT; VAREJÃO, 2015). Nevertheless, this validation method might be also unfair, because the evaluated methods probably do not use the same data division. Thus, a fairer validation should use the same folds division for all classification methods tested. It is called paired cross-validation. Then, a proper validation method for non-parametric classification methods and datasets with no explicit division of training and test could be a repeated paired cross-validation, e.g. ten rounds of paired 10-folds cross-validation. Each round with different folds division but the same folds division for all methods.

Another problem emerges when parametric classification methods are tested. These methods need a tuning phase that cannot use the whole training dataset. Otherwise, it could generate an over-optimistic biased result. Therefore, a nested cross-validation is necessary. In the nested cross-validation the folds are divided in training and test, then part of the training dataset is divided again in training and validation. For instance, let us assume that the outer validation is a 3-fold cross-validation. The first test would used the first and the second folds as training dataset and the third fold as test. The tuning phase must use only the training dataset, i.e. first + second folds. Thus, the classification methods might use the first fold for training and the second for validation. However, this approach would have great chance of over-fitting. Then, a 2-fold cross validation could be applied in the inner validation. Inner validation is that used to tuning the classification method. Or, a single hold-out for inner validation could also be appropriated, since the inner training and inner validation datasets does not must have a fixed division. Figure 10 presents an example of nested cross-validation. In the example, a 3-fold cross-validation is used as outer-validation and a 4-fold cross-validation as inner-validation. As stated before, the outer-validation and the inner-validation do not need to be equal. For instance, it would spend an excessive and unnecessary time to run ten rounds of 10-fold cross-validation to tune the classifier parameters. Or worse, applying a feature selection methods as wrapper, to evaluate each feature subset using such an expensive validation method. Therefore, all feature selection methods applied as wrapper in this work used the average of five holdouts validation to evaluate each subset of features, providing a good trade off between time and classification performances.

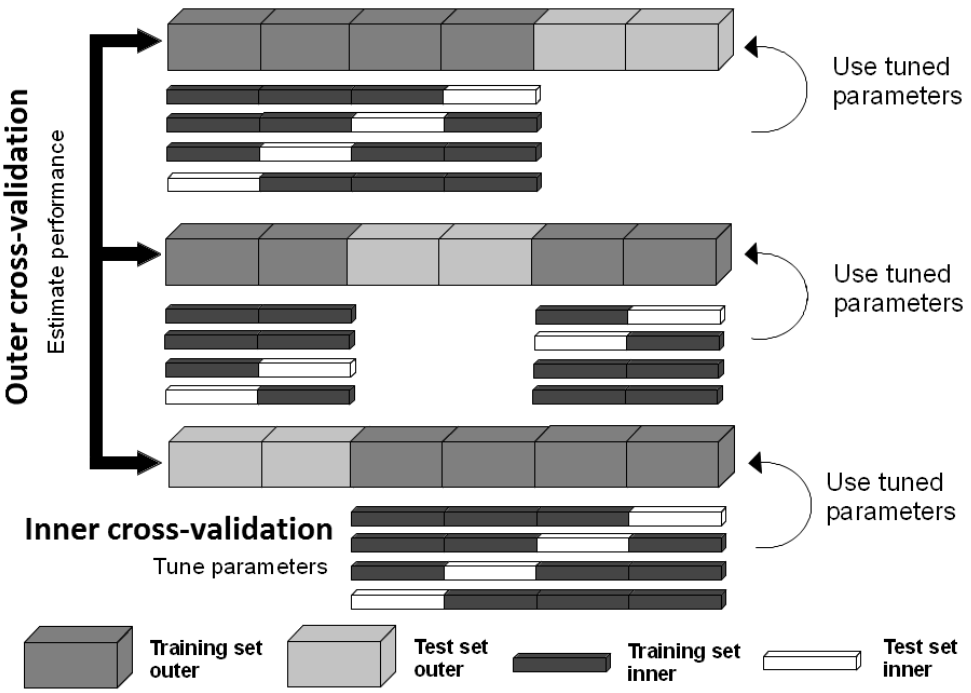


Figure 10 – Nested cross-validation

In the present work, a multiple paired nested cross-validation was performed, as done in OLIVEIRA-SANTOS et al. (2016); BOLDT et al. (2017); RAUBER et al. (2017). It means that all evaluated methods used the same data division for training and test. Despite this validation method provides no guarantee that one method is really better than other, it ensures that both methods use equal dataset divisions. Therefore, this kind of paired validation tends to reduce the probability of Type I error (DEMŠAR, 2006). A Type I error is a conclusion of an experiment that there is a significative difference between algorithms, while in reality there is none.

2.2.2 Metrics

Currently, accuracy still is the most used classification performance metric used for fault diagnosis. However, due its limitations, it is gradually losing space in the literature. Its main limitation is the treatment for unbalanced datasets, that includes the majority of the industrial process datasets used in fault diagnosis. Yet, accuracy is also used in this work due its popularity. Considering the classes of table 1, the accuracy is estimated as

$$Accuracy = \frac{tp + tn}{tp + fn + fp + tn} \tag{2.6}$$

When the number of sample of each class is equal (or approximately equal) the dataset is considered balanced. Then, unbalanced datasets are those which have different amounts of samples for each class. For datasets with unbalanced number of samples for each class, there are more appropriate choices to measure the classification performance than accuracy. Following, some metrics that are more appropriated for unbalanced datasets.

2.2.2.1 F-measure

F-measure, is a better metric for unbalanced datasets than accuracy because it is sensitive to false positives and false negatives simultaneously. The F-measure estimation is a combination of two other metrics, precision and recall (SOKOLOVA; LAPALME, 2009). The estimations of precision, recall and F-measure for binary problems are presented in equation 2.7, 2.8 and 2.9, respectively.

$$Precision = \frac{tp}{tp + fp} \tag{2.7}$$

$$Recall = \frac{tp}{tp + fn} \tag{2.8}$$

$$F\text{-}measure = \frac{(\lambda^2 + 1)tp}{(\lambda^2 + 1)tp + (\lambda^2)fn + fp} \tag{2.9}$$

In equation 2.9, λ is a weighting parameter. When λ is set to one ($\lambda = 1$) the weight given for precision and recall is the same. In this case, the F-measure is also known as F1-measure or F1-score. When the parameter λ is set to a value smaller than one ($\lambda < 1$), it favors recall, while it is set to a value bigger than one ($\lambda > 1$), it favors precision.

According to SOKOLOVA; LAPALME (2009), for multi-class datasets F-measure macro-averaged ($F\text{-}measure_M$) is the best choice, because macro-averaging treats all classes equally while micro-averaging favors bigger classes. Its estimation (equation 2.12) is based on the precision macro-averaged (equation 2.10) and the recall macro-averaged (equation 2.11), where l is the number of labels. In equation 2.12, λ has the same meaning as in equation 2.9.

$$Precision_M = \frac{\sum_{i=1}^{l} \frac{tp_i}{tp_i + fp_i}}{l} \tag{2.10}$$

$$Recall_M = \frac{\sum_{i=1}^{l} \frac{tp_i}{tp_i + fn_i}}{l} \tag{2.11}$$

$$F\text{-}measure_M = \frac{(\lambda^2 + 1) Precision_M Recall_M}{\lambda^2 Precision_M + Recall_M} \tag{2.12}$$

2.2.2.2 G-mean

Suggested in KUBAT; MATWIN et al. (1997), the Geometric Mean (G-mean) is another confusion matrix based metric able to deal with unbalanced datasets. It takes into account the sensitivity and the specificity. The estimation of sensitivity, specificity and G-mean is given by the equations 2.13, 2.14 and 2.15, respectively.

$$Sensitivity = Recall = \frac{tp}{tp + fn} \tag{2.13}$$

$$Specificity = \frac{tn}{tn + fp} \tag{2.14}$$

$$G\text{-}mean = \sqrt{Sensitivity \times Specificity} \tag{2.15}$$

Unlike the F-measure, the G-mean measure does not have a parameter to weight the importance of sensitivity or specificity. Multi-class problems can use the average of the individual G-mean for each class.

2.2.2.3 Receiver Operating Characteristic

Beyond these confusion matrix based performance metrics presented, the Receiver Operating Characteristic (ROC) and the area under the ROC curve (AUC-ROC) (DAVIS; GOADRICH, 2006) are often used to measure classification in the fault diagnosis field, specially for imbalanced datasets. The ROC is a graph showing the relation between the sensitivity 2.13 and the specificity 2.14, as the decision threshold varies. Figure 11 shows a hypothetical comparison between two algorithms using ROC curves. ROC have been used for a long time to compare unbalanced datasets. However, it has been losing popularity while metrics like F-measure has been more used. One motivation for this change is that ROC needs the label and the confidence level of a classifier to be calculated. Also, the graph generation spend time that might slow down nested validations. The AUC-ROC summarizes the classification performance in a single value. Higher areas indicate better classification.

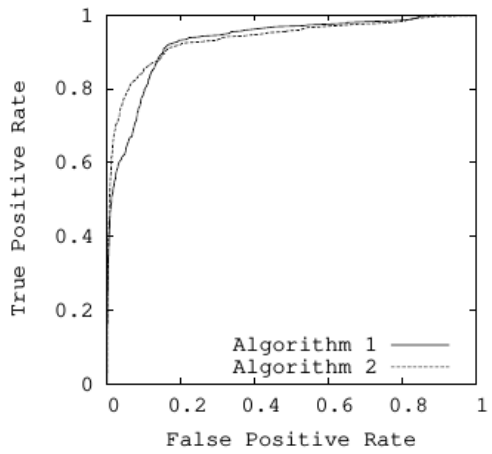


Figure 11 – ROC curves comparing two algorithms (DAVIS; GOADRICH, 2006).

All metrics presented in this section are used in this work. Chapter 4 uses accuracy, AUC-ROC and G-mean; chapter 5 uses accuracy, precesion, recall and F-measure; chapter 6 uses accuracy, precision, recall, F-measure and G-mean.

2.2.3 Statistical Tests

After estimate some metric through a validation method for comparing classification approaches, it is interesting to know how significantly different these methods are. Two statistical tests are used here. Both of them estimate the statistical significance of a method over other by the value of some metric for each fold of multiple paired cross-validations.

The statistical test presented in OLIVEIRA-SANTOS et al. (2016)was designed to estimate the statistical significant difference between two methods for a single dataset. Correlated t test (NADEAU; BENGIO, 2003) is used to assess whether the mean difference

between each pair of classifiers are significantly different. Holm's procedure (HOLM, 1979) is used to explain the increase of the Type I error (i.e. rejecting the hypothesis that the classifiers performance are similar, when they actually are) involved in drawing conclusion from multiple hypothesis tests. The statistics of the correlated t test proposed by NADEAU; BENGIO (2003) is given by (2.16)

$$t = \frac{\overline{d}}{\sqrt{s^2 \left(\frac{1}{R \times K} + \frac{n^{\mathrm{TE}}}{n^{\mathrm{TR}}}\right)}} \tag{2.16}$$

where $\overline{d}$ and s^2 are the mean and variance of the differences between measures of two different methods, respectively. K is the number of folds for the cross-validation method and R is the number of rounds that this cross-validation was repeated. The quantities n^{TR} and n^{TE} are the size of the training and test sets, respectively. Under the assumption that the classifiers have similar performances, the hypothesis of similarity between classifier can be rejected if the probability of a standard Student's t-distribution with $R \times K - 1$ degrees of freedom (T) being greater than the absolute value of the observed t (from 2.16) is smaller than a chosen significance level α, i.e. $P(T > |t|) < \alpha$. This probability is the p-value of the correlated t test calculated by Holm's step-down procedure (HOLM, 1979). This procedure sort the p-values of the associated hypotheses so that $p_1 < p_2 < \ldots < p_q$, where q is the total number of comparisons. The largest i such that $p_i < \alpha/(q + 1 - i)$, called i^*, is used to reject the hypothesis of the tests with the following p-values $p_1, \ldots, p_{i^*}$.

Unlike OLIVEIRA-SANTOS et al. (2016), the method presented in CORANI et al. (2016) is designed to present the probability of one classifier be statistically better, equal or worse then other, over multiple datasets. Considering a collection of q data sets, the actual mean difference of measure on the i-th dataset is δ_i. On the i-th data set, the cross-validation measures $x_{i1}, x_{i2}, \ldots, x_{in}$ are cross-correlated with correlation ρ because of the overlapping training sets built during cross-validation. Their sample mean and standard deviation are $\bar{x}_i$ and s_i. The Maximum Likelihood Estimator (MLE) of δ_i is $\bar{x}_i$. The average difference of some measure on the population of data sets is δ_0. The Bayesian hierarchical model proposed by CORANI et al. (2016) has the following probabilistic model as assumption, where MVN means Multivariate Normal Distribution and "unif" means Uniform Distribution:

$$\mathbf{x}_i \sim MVN(\delta_i, \sigma_i) \tag{2.17}$$

$$\delta_i \ldots \delta_q \sim t(\delta_0, \sigma_0, \nu) \tag{2.18}$$

$$\sigma_i \ldots \sigma_q \sim \mathrm{unif}(0, \bar{\rho}) \tag{2.19}$$

Equation 2.17 models the fact that the cross-validation measures $x_{i1}, x_{i2}, \ldots, x_{in}$ of the i-th dataset are jointly generated from random variables which are equally cross-correlated (ρ), which have the same mean (δ_i) and variance (σ_i) and which are jointly normal. Equation 2.18 models the fact that the mean difference of performances in the single datasets, δ_i, depends on δ_0 that is the "ground truth" difference between the classifiers. The hierarchical model assigns to the i-th data set its own standard deviation σ_i, assuming the σ_i's to be drawn from a common distribution (equation 2.19). The limits of the uniform distribution of $\sigma_0 \sim \text{unif}(0, \bar{\sigma}_0)$ are suitable to deal with indicators whose difference bounded between 1 and -1, such as accuracy, AUC, precision, recall, F-measure and G-mean.

2.3 Feature Selection

Section 2.1 stated that the samples of some dataset are informed to a classifier, during its training or its test, throw features. The given example showed that the number of angles of stars and pentagons could be considered a good feature to discriminate these geometric figures while their color not. Unfortunately, in real problems, including fault diagnosis problems, identifing whether a feature produces good discrimination among the classes is not always an easy task. This study area is known as feature selection.

Feature selection can be seen as a combinatorial optimization problem, composed of a selection criterion and a search strategy, improving prediction performance, and reducing problem dimensionality (GUYON; ELISSEEFF, 2003). Since an exhaustive search might be computationally unfeasible, suboptimal strategies are employed in general (RAUBER; BOLDT; VAREJÃO, 2015). Feature selection algorithms, both optimal (for relatively small data sets) and suboptimal (in general), commonly are used as filter, wrapper or embedded methods. Filter methods calculate the quality of the combination of some features based on data only, so they are faster than wrapper methods, which need a classifier to evaluate a feature combination according to some validation method. Wrapper methods are considered more sophisticated (PENG; LONG; DING, 2005) and usually achieve better results (KUDO; SKLANSKY, 2000) than filter methods, with the drawback of a higher computational burden. Embedded algorithms, made for specific classifiers, can be faster than wrapper methods, without loss of quality. On the other hand, embedded methods do not work for any classifier as wrapper methods do.

2.3.1 Ranking Feature Selection

Ranking algorithms provide a univariate evaluation of the features without considering possible mutual dependencies among them (BOLDT; RAUBER; VAREJÃO, 2015). The feature evaluation can be done using only data, when a ranking algorithm is applied

as a filter, or using a classifier with some performance estimation, when it is applied as a wrapper. The ranking algorithm receives the initial feature set $\mathcal{F}$ and returns a list $\mathcal{R}$. The list $\mathcal{R}$ contains the features of $\mathcal{F}$ sorted in descendant order with respect to their individual quality. The final feature subset $\mathcal{G}$ has the n highest ranked features, where n is the number of features to be selected. Usually, ranking algorithms are faster and have inferior performance than multivariate methods, which consider mutual dependence among the features. Nevertheless, many feature selection algorithms include feature ranking as the principal or auxiliary selection mechanism because of its simplicity, scalability, and good empirical results (STAŃCZYK, 2015; BOLDT; RAUBER; VAREJÃO, 2015). Figure 12 illustrates a feature selection scheme based on ranking algorithm. The features are evaluated one by one, according to some criterion, then reordered according to this criterion. The evaluation criterion for each feature can be calculated in parallel or sequentially, as the different arrows sizes of the initial feature set in figure 12 indicate. Finally, the first n features are selected. For the example of figure 12, n is set to 3. The feature selection based on the ranking algorithm was used in BOLDT; RAUBER; VAREJÃO (2017) and (BOLDT et al., 2017). These works are also presented in chapter 5 and 6, respectively.

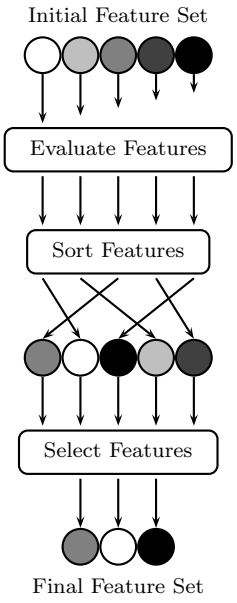


Figure 12 – Feature Selection based on Ranking Algorithm.

2.3.2 Sequential Feature Selection

Sequential Forward Selection (SFS) is a well known multivariate suboptimal feature selection based on a sequential greedy search algorithm, which is a good compromise between exploration of the search space and computational cost. In order to select n of the $|\mathcal{F}|$ features from the complete set $\mathcal{F}$, SFS initializes with an empty feature set $\mathcal{G} \leftarrow \emptyset$. Features are iteratively added to $\mathcal{G}$, according to some selection criterion. The algorithm stops and returns $\mathcal{G}$ when $|\mathcal{G}| = n$. SFS runs a complete performance estimation

for each remaining candidate feature $f_j \in \mathcal{F}, f_j \notin \mathcal{G}$ jointly with the already selected set $\{f_j \cup \mathcal{G}\}$, $j \in \{1, \ldots, |\mathcal{F}|\}$. Those features that mostly increase, or less decrease the selection criterion are joined to $\mathcal{G}$. Ties are solved arbitrarily. Figure 13 illustrates how SFS works for $n = 3$. It can be seen that SFS has a higher complexity than the ranking algorithm. Even though, SFS is considered a good trade-off between speed and quality. Usually, SFS produces better results than ranking, because it evaluates subsets of features instead of isolated features. As it makes more comparisons, SFS is also slower than ranking.

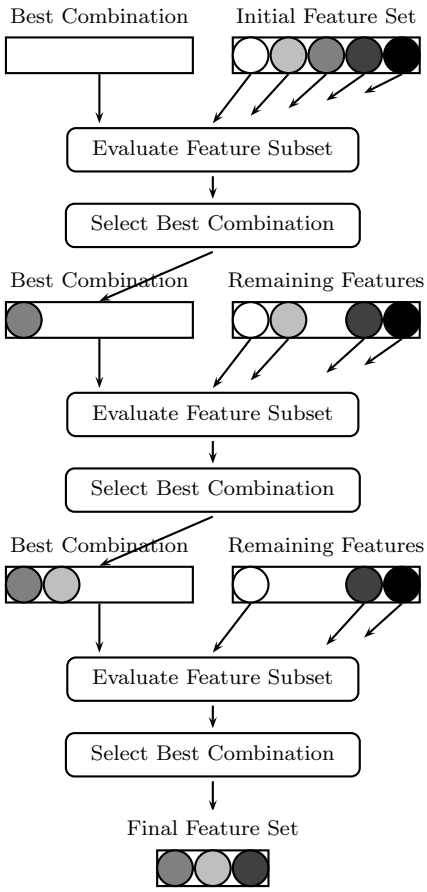


Figure 13 – Feature Selection based on Sequential Forward Selection Algorithm. Initially the Best Combination is an empty set.

Like ranking methods, SFS is also used as the principal or auxiliary selection mechanism. One example is the minimal Redundancy Maximal Relevance (mRMR) feature selection algorithm (PENG; LONG; DING, 2005) that is based on mutual information. It can be divided into two phases. In its first phase, the SFS algorithm is applied as a filter, using the minimal-redundancy-maximal-relevance criterion to analyze all features in the data set. Thus, the SFS algorithm actually returns a multivariate ranking of all $|\mathcal{F}|$ features. The second phase uses a wrapper approach trying to find the best number of features $n \leq |\mathcal{F}|$. This choice is made testing $|\mathcal{F}|$ feature sets with different sizes, increasing them sequentially according to the returned rank. For instance, assume that the original feature set is $\mathcal{F} = \{1, 2, 3, 4\}$ and the SFS algorithm returns the ranking $\mathcal{R} = \{3, 4, 1, 2\}$. The

algorithm will choose the subset with highest performance among $\mathcal{G}_1 = \{3\}$, $\mathcal{G}_2 = \{3, 4\}$, $\mathcal{G}_3 = \{3, 4, 1\}$, $\mathcal{G}_4 = \{3, 4, 1, 2\}$. Such algorithms that mix approaches like ranking, filter and wrapper are usually called hybrid algorithms (BOLDT et al., 2015; BOLDT; RAUBER; VAREJÃO, 2017).

Another sequential algorithm is the Sequential Backward Selection (SBS) that works like SFS, but removes features initially from the total set, instead of increasing an initial empty set. Commonly SBS is slower than SFS. For instance, to select x features from a dataset with n features, SFS starts doing n wrapper evaluations and SBS also. Although, the SFS evaluations use classifiers with one feature while SBS evaluations use classifiers with $n - 1$ features. As more features are used, evaluation becomes slower. Both algorithms, SFS and SBS, can be applied as filter or wrapper methods. However, these algorithms are slower and achieve better results when they are used as wrapper methods.

Floating techniques (PUDIL; NOVOVIČOVÁ; KITTLER, 1994) allow backtracking for an arbitrary number of times as long as the quality criterion J is improving. As representatives for a complete sequential search strategy algorithm, the Sequential Floating Forward Search (SFFS) and Sequential Floating Backward Search (SFBS) are used in chapter 4.

2.3.3 Selecting Features using Heuristics and Metaheuristics

Since feature selection can be seen as a combinatorial problem, it can be approximated with any heuristic or metaheuristic designed for combinatorial problems. In fact, the SFS and SBS algorithms are based on greedy heuristics. As well, SFFS and SFBS are also heuristics. At all events, any heuristic can be used to select features, e.g. down hill, multi-start. Thus, metaheuristics like Genetic Algorithm (GA) (TSAI; EBERLE; CHU, 2013), Particle Swarm Optimization (PSO) (YONG; DUN-WEI; WAN-QIU, 2016) and Ant Colony Optimization (ACO) (ROUHI; NEZAMABADI-POUR, 2016) can be used for feature selection. The adaptation of a metaheuristic to perform feature selection can be done considering features as a binary vector, where true means presence of a feature and false its absence. The objective function evaluates the vector according to the feature selection criterion, that can be either filter or wrapper. For the filter case, the objective function needs only the dataset and the binary vector to calculate its fitness. For the wrapper case, it is also necessary a classifier and a validation method as parameters. This work uses a feature selection method based on the GA implementation in the Matlab Global Optimization Toolbox. Usually, the GA based feature selection outperforms the SFS and ranking algorithms. Preliminary experiments also used a PSO based feature selection, but it does not surpass the GA version. Therefore, only the GA related feature selection results are juxtaposed in chapter 5.

2.4 Classifier Ensembles

The combination of classifiers, also known as classifier ensembles, is a well established approach (WANDEKOKEN et al., 2011). This approach consists in use simultaneously divergent accurate classifiers. The term accurate classifier used here refers to a classifier that achieves more than 50% of accuracy in tests. The term divergent means that the classifiers give their wrong prediction in a different region of the global feature space. In other words, the classifiers that compound the ensemble give the wrong prediction to different test samples. Figure 14 shows how a classifier ensemble can improve its classification performance combining divergent classifiers.

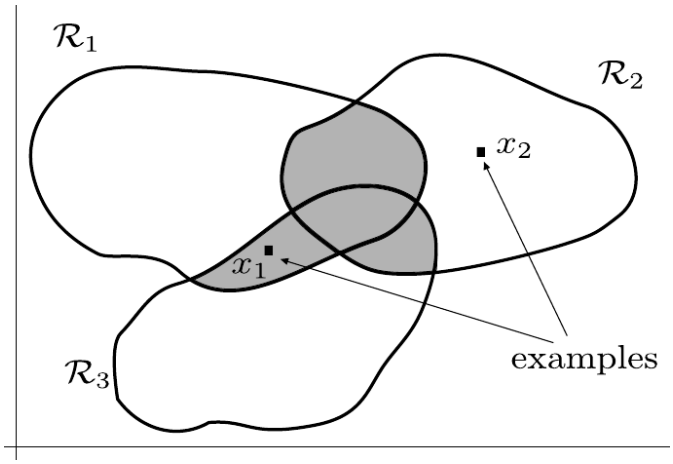


Figure 14 – The intersection of the individual wrong decision is necessarily smaller than the region of wrong decisions of any individual component classifiers (the regions R1 , R2 and R3) (WANDEKOKEN et al., 2011).

2.4.1 Bagging, Adaboost and Random Forest

One of the earliest ensemble approaches is **b**ootstrap **agg**regat**ing** (Bagging) (BREIMAN, 1996). The diversity among the classifiers that compose the ensemble is obtained using different datasets to train each base classifier. These datasets are generated with the statistical technique bootstrap. The bootstrap eliminates some samples and replicate others, keeping the same number of samples of the original dataset. Bagging generates ensembles with low trend of over-fitting (QUINLAN, 1996). Instead of drawing a succession of independent bootstrap samples from the original instances, the Adaboost algorithm (FREUND; SCHAPIRE, 1996) maintains a weight for each instance. The instance weight is directly proportional to its influence in the classifier learning. At each trial, the vector of weights is adjusted reflecting the performance of the classifier. The weight of misclassified instances is increased in each iteration. The final classifier also aggregates the learned classifiers by voting. However, the vote of each classifier is a function of its accuracy. Despite its popularity, Adaboost has highest trends of over-fitting than Bagging (QUINLAN, 1996). Unlike Bagging and Adaboost, Random Forests (BREIMAN,

2001) raffles samples and features simultaneously to generate divergent tree classifiers to compose the final ensemble. Random Forests have shown good results and have been widely cited in the literature.

2.4.2 Ensemble Feature Selection

The two main ways to promote diversity among the classifiers is selecting samples or selecting features. Many works use feature subset variations to promote the necessary diversity in a classifier ensemble (WANDEKOKEM et al., 2011; OPITZ, 1999; BREIMAN, 2001; DIAO et al., 2014). In these works, the different subsets were obtained using feature selection algorithms, instead a pure random choice as performed in Random Forests. In WANDEKOKEM et al. (2011), the method varies the hyperparameters (kernel type, kernel parameter) of Support Vector Machine (SVM) (BURGES, 1998) to generate different feature subsets, using the Sequential Forward Selection (SFS) (GUYON; ELISSEEFF, 2003) algorithm. The SVMs trained with the generated subsets are combined to form an ensemble. Redundant SVMs are removed by Sequential Backward Selection (SBS) search. The SBS algorithm works like SFS, but removes features initially from the total set X, instead of increasing an initial empty set. However, in BSFS case, it removes SVMs instead features. A similar idea like the last part of BSFS, when SBS is applied to reduce the number of SVMs, is used in DIAO et al. (2014), where ensemble predictors are transformed into training samples and classifiers are treated as features. This process aims to reduce the run-time overhead of the system, while improving the overall efficiency.

The Genetic Ensemble Feature Selection (GEFS) algorithm proposed in OPITZ (1999) is inspired by the Genetic Algorithm (GA) paradigm to select feature subsets in a wrapper approach. The representation of each individual of the GEFS population is an array of integers, where each integer indexes a feature. GEFS calculates the fitness of each individual i as: $Fitness_i = Accuracy_i + \lambda Diversity_i$, where λ defines the trade-off between accuracy and diversity, and its values variate between zero and one. The algorithm prunes the population to the N most-fit individuals, updates the λ parameter, then repeats this process. At every generation, the current ensemble consists of averaging the predictions of the current population. The main problem of GEFS is exactly the use of GA, because GA causes all individuals to converge towards a single point in the solution space. As the ensemble quality depends on the diversity of its classifiers, GEFS forces a diversity in its objective function. Hence, the GA fitness function is not the same function to be maximized, that is the ensemble accuracy. Thus, GEFS works on a dilemma controlled by the parameter λ. If the λ value goes to zero, all individuals trends to converge to a single point and there is no need to make an ensemble, since all answers will be the same. As the λ goes to one, the guarantee of the individuals quality reduces. Therefore, GEFS adjusts λ to maximize the ensemble accuracy.

More recent work (PARK; KIM, 2015) presents the Sequential Random K-Nearest Neighbor (SRKNN) feature selection algorithm. The SRKNN algorithm is, in a certain way, similar to random forests, since each base classifier is modeled based on a sequential forward selection strategy from a bootstrapped sample, and it uses a majority voting scheme. The main difference between SRKNN and random forests is that the first one uses a k-NN (COVER; HART, 1967) classifier instead of decision trees to form the ensemble.

In GHIMIRE; LEE (2014) ensembles of ELM were made by using a bagging algorithm for Facial Expression Recognition (FER). Their method consists in extract the features from the face image by dividing it into a number of small cells. Then, a bagging algorithm was used to construct many different bags of training data and each of them was trained by using separate ELMs. To recognize the expression of the input face image, HOG features were fed to each trained ELM and the results were combined by using a majority voting scheme. The ELM ensemble using bagging improved the generalized capability of the network significantly.

More recently, CAO; CHEN; FAN (2016) proposed a majority voting based ELM ensemble for landmark recognition application to be used in mobile devices. The ELM was chosen as base classifier because it has fast training and good classification performance. It was considered the best choice to enhance the landmark recognition performance and maintain the training and recognition time at an acceptable level. No sophisticated technique was applied to training the ensemble that was composed by k ELMs trained with all samples and features. Only the unstable nature of the random weights in the hidden layer of the ELMs was used to generate the desirable divergence among the classifiers.

Unlike CAO; CHEN; FAN (2016), HAN; LIU (2015) promotes the diversity within the ensemble, adopting feature segmentation and then feature extraction with nonnegative matrix factorization to the original data firstly. The authors argued the ELM was chosen as base classifier to improve the classification efficiency. According to them, the experimental results showed that the ensemble not only had high classification accuracy, but also handled the adverse impact of a few of labeled training samples in the classification.

3 Industrial Processes

This work uses three industrial processes as case study. The first process is the Case Western Reserve University Bearing Data. It is a widely cited process composed by a large number of labeled vibratory data in Matlab format. Normal and faulty rolling bearing conditions can be identified with this data. The second process is the Tennessee Eastman chemical process. It uses a simulator to generate the data that is public available. The last process is related with Electrical Submersible Pumps. The data from this process was obtained by a partnership between the Ninfa (Núcleo de Inferência e Algoritmos) Laboratory and Petrobras (Rio de Janeiro - Brazil).

3.1 Case Western Reserve University Bearing Data

The bearing dataset provided by the Bearing Data Center of Case Western Reserve University (CWRU) (CWRU, 2017) is a public available vibratory data widely cited in the scientific literature (RAUBER; BOLDT; VAREJÃO, 2015; XIA et al., 2012; LIU, 2012; WU et al., 2012). Its public availability allows scientists to compare their research to others in the literature. The dataset is composed by vibratory signals of normal and fault bearings extracted from a 2 hp reliance electric motor. The faults were introduced at a specific position of the bearing, using an electro-discharging machining with fault diameters of 0.007, 0.014, 0.021 and 0.028 inches. A dynamometer induced loads of 0, 1, 2 and 3 hp, changing the shaft rotation from 1797 to 1720 rpm. One model of bearing was used on the drive end and the other was used on the fan end. Tables 3 and 4 show, for each bearing, the dimensions and the frequencies where each fault is manifested, respectively.

Table 3 – Bearing dimensions (inches). The models names are reduced. Their complete name is 620?-2RS JEM SKF, deep groove ball bearing, where ? means 5 or 3 for each model.

Position	Model	Inside Diameter	Outside Diameter	Thickness	Ball Diameter	Pitch Diameter
Drive end	6205-2RS	0.9843	2.0472	0.5906	0.3126	1.537
Fan end	6203-2RS	0.6693	1.5748	0.4724	0.2656	1.122

Table 4 – Bearing defect frequencies: (multiple of running speed in Hz). The models names are reduced. Their complete name is 620?-2RS JEM SKF, deep groove ball bearing, where ? means 5 or 3 for each model.

Position	Model	Inner Ring	Outer Ring	Cage Train	Rolling Element
Drive end	6205-2RS	5.4152	3.5848	0.39828	4.7135
Fan end	6203-2RS	4.9469	3.0530	0.3817	3.9874

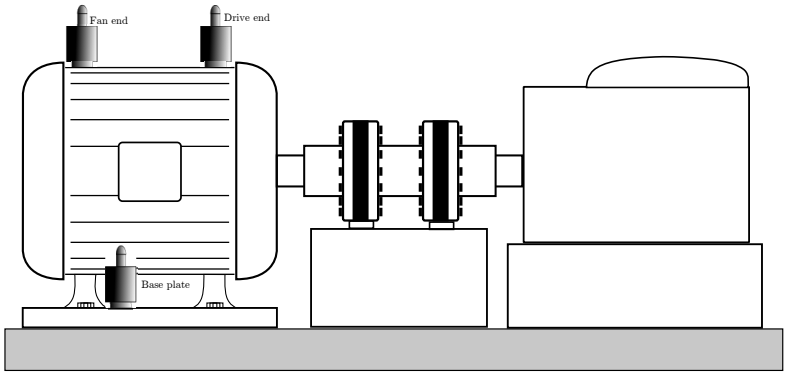


Figure 15 – CWRU test bed for bearing faults diagnosis.

Three accelerometers collected the vibratory data, placed on the drive end, fan end and the base of the motor, as shown in figure 15.

3.1.1 Identified Classes

The identified classes can be labeled according to the number of bearings, the bearing state (normal or defective), fault location (when the bearing is faulty), fault severity (depth) and motor load. Table 5 presents the distribution and description of the classes used in the experiments performed in RAUBER; BOLDT; VAREJÃO (2015). It is possible to identify not only the fault class, but within the same class also the severity of the fault. (XIA et al., 2012; LIU, 2012; WU et al., 2012) identify only a small number of classes, from one sensor position (drive end). (XIA et al., 2012) uses four classes: normal, ball, inner race and outer race, or fixes a fault class and then distinguishes among its severities. Despite the existence of examples for the failures in the Ball and the Inner Race with 0.028 inches in the drive end bearing, these examples do not have the vibratory data of the fan end bearing. Then, these failures are usually ignored.

In RAUBER; BOLDT; VAREJÃO (2015) the faults presented in table 6 are also identified. Since the ROC-AUC of the CWRU is very high for many classifiers using heterogeneous feature methods, the condition with smaller hit rate (BALL_FE) was chosen to be identified by its load. Thereby, it was possible to verify differences among techniques and classifiers.

3.1.2 Feature Extraction Models

Vibratory signals, collected by accelerometers, are widely used in automatic rotating machine failure diagnosis. The next three subsections present a set of representative feature extraction techniques that were used in RAUBER; BOLDT; VAREJÃO (2015). Statistical models are applied in the time and frequency domain, while wavelet package analysis represents an extraction in the time-frequency domain (XIA et al., 2012). Complex envelope analysis completes the methods used in the frequency domain.

Table 5 – Class distribution and description for the CWRU bearing dataset.

Class	Name	Samp	Distr	Description
1	Ball_DE_007	120	4.97 %	0.007” ball fault in the drive end.
2	Ball_FE_007	60	2.48 %	0.007” ball fault in the fan end.
3	Ball_DE_014	120	4.97 %	0.014” ball fault in the drive end.
4	Ball_FE_014	60	2.48 %	0.014” ball fault in the fan end.
5	Ball_DE_021	120	4.97 %	0.021” ball fault in the drive end.
6	Ball_FE_021	60	2.48 %	0.021” ball fault in the fan end.
7	InnerRace_DE_007	120	4.97 %	0.007” inner race fault in the drive end.
8	InnerRace_FE_007	60	2.48 %	0.007” inner race fault in the fan end.
9	InnerRace_DE_014	120	4.97 %	0.014” inner race fault in the drive end.
10	InnerRace_FE_014	60	2.48 %	0.014” inner race fault in the fan end.
11	InnerRace_DE_021	120	4.97 %	0.021” inner race fault in the drive end.
12	InnerRace_FE_021	60	2.48 %	0.021” inner race fault in the fan end.
13	Normal	60	2.48 %	Normal
14	OuterRace_DE_007	360	14.91 %	0.007” outer race fault in the drive end.
15	OuterRace_FE_007	180	7.45 %	0.007” outer race fault in the fan end.
16	OuterRace_FE_014	75	3.11 %	0.014” outer race fault in the drive end.
17	OuterRace_DE_014	120	4.97 %	0.014” outer race fault in the fan end.
18	OuterRace_DE_021	360	14.91 %	0.021” outer race fault in the drive end.
19	OuterRace_FE_021	60	2.48 %	0.021” outer race fault in the fan end.

Table 6 – Classes of the special AUC-ROC dataset, varying fault severity and load for the fan end ball fault (RAUBER; BOLDT; VAREJÃO, 2015).

Class	Name	Samples	Distribution	Description
1	Ball_FE_0_007	50	8.33%	0.007”, 0 hp load
2	Ball_FE_1_007	50	8.33%	0.007”, 1 hp load
3	Ball_FE_2_007	50	8.33%	0.007”, 2 hp load
4	Ball_FE_3_007	50	8.33%	0.007”, 3 hp load
5	Ball_FE_0_014	50	8.33%	0.014”, 0 hp load
6	Ball_FE_1_014	50	8.33%	0.014”, 1 hp load
7	Ball_FE_2_014	50	8.33%	0.014”, 2 hp load
8	Ball_FE_3_014	50	8.33%	0.014”, 3 hp load
9	Ball_FE_0_021	50	8.33%	0.021”, 0 hp load
10	Ball_FE_1_021	50	8.33%	0.021”, 1 hp load
11	Ball_FE_2_021	50	8.33%	0.021”, 2 hp load
12	Ball_FE_3_021	50	8.33%	0.021”, 3 hp load

3.1.2.1 Statistical Model

This section presents ten statistical features in the time domain and three in the frequency domain. As a representative set we choose those features proposed in XIA et al. (2012),c.f. table 7 and table 8. Table 7 presents the definition of statistical features in the time domain as root mean square (RMS), square root of the amplitude (SRA), kurtosis value (KV), skewness value (SV), peak-peak value (PPV), crest factor (CF), impulse factor (IF), margin factor (MF), shape factor (SF) and kurtosis factor (KF). Table 8

presents the definition of statistical features in the frequency domain as frequency center (FC), RMS frequency (RMSF) and root variance frequency (RVF), where f_i is the time domain value transformed to the frequency domain using a Fast Fourier Transform (FFT).

Table 7 – Time domain statistical feature set of the vibration signal.

$X_{\text{rms}} =$	$\left(\frac{1}{N}\sum_{i=1}^{N} x_i^2\right)^{1/2}$	$X_{\text{sra}} =$	$\left(\frac{1}{N}\sum_{i=1}^{N} \sqrt{\lvert x_i\rvert}\right)^{2}$
$X_{\text{kv}} =$	$\frac{1}{N}\sum_{i=1}^{N} \left(\frac{x_i - \overline{x}}{\sigma}\right)^{4}$	$X_{\text{sv}} =$	$\frac{1}{N}\sum_{i=1}^{N} \left(\frac{x_i - \overline{x}}{\sigma}\right)^{3}$
$X_{\text{ppv}} =$	$\max(x_i) - \min(x_i)$	$X_{\text{cf}} =$	$\frac{\max(\lvert x_i\rvert)}{\left(\frac{1}{N}\sum_{i=1}^{N} x_i^2\right)^{1/2}}$
$X_{\text{if}} =$	$\frac{\max(\lvert x_i\rvert)}{\frac{1}{N}\sum_{i=1}^{N} \lvert x_i\rvert}$	$X_{\text{mf}} =$	$\frac{\max(\lvert x_i\rvert)}{\left(\frac{1}{N}\sum_{i=1}^{N} \sqrt{\lvert x_i\rvert}\right)^{2}}$
$X_{\text{sf}} =$	$\frac{\max(\lvert x_i\rvert)}{\left(\frac{1}{N}\sum_{i=1}^{N} x_i^2\right)^{1/2}}$	$X_{\text{kf}} =$	$\frac{\frac{1}{N}\sum_{i=1}^{N} \left(\frac{x_i - \overline{x}}{\sigma}\right)^{4}}{\left(\frac{1}{N}\sum_{i=1}^{N} x_i^2\right)^{2}}$

Table 8 – Frequency domain statistical feature set of the vibration signal.

$X_{\text{fc}} =$	$\frac{1}{N}\sum_{i=1}^{N} f_i$	$X_{\text{rmsf}} =$	$\left(\frac{1}{N}\sum_{i=1}^{N} f_i^2\right)^{1/2}$
$X_{\text{rvf}} =$	$\left(\frac{1}{N}\sum_{i=1}^{N} (f_i - X_{\text{fc}})^2\right)^{1/2}$		

This extraction model has the advantage of not requiring any parameters, its calculus is simple and has a low computational cost. Its main drawback is its inferior capacity of class identification when compared to more complex methods. However, these features, used together with others extracted by complementary methods, followed by feature selection, can help improving the classification system, as shown in RAUBER; BOLDT; VAREJÃO (2015). The total number of statistical features extracted from the CWRU data is usually $(10 + 3) \times 2 = 26$, i.e. the ten statistical features of the time domain, the three of the frequency domain, taken by two accelerometers at the fan end and drive end accelerometers. The base plate accelerometer is not present in all vibration

files, therefore, its data is discarded in order to generate all samples with the same number of features.

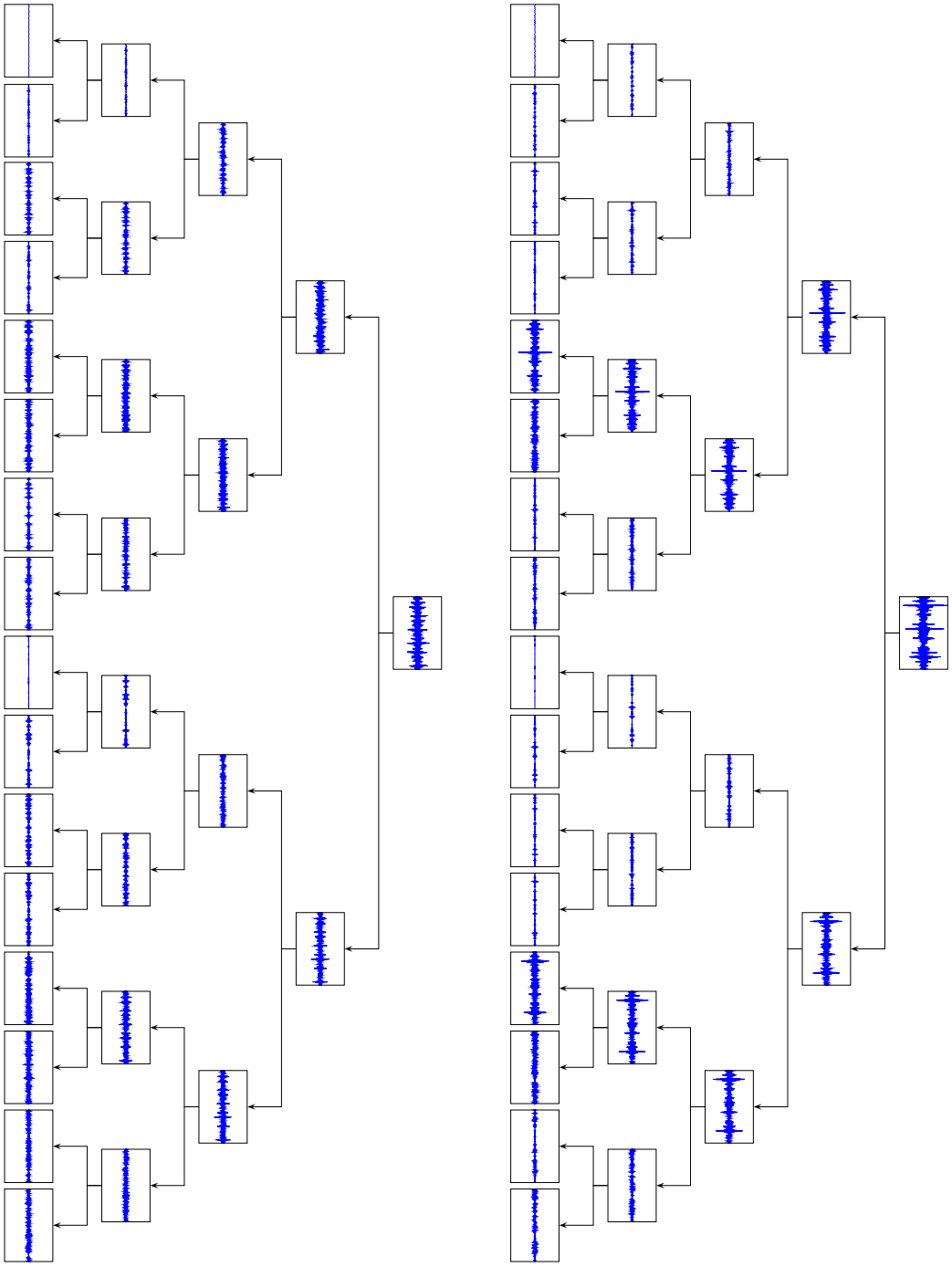
3.1.2.2 Wavelet Package Analysis

Dual domain analysis methodologies that extract features from the time-frequency representation are represented in this work by wavelets (GAO; YAN, 2010). Posterior to the classical wavelet decomposition, the set of vibration analysis techniques has been enriched by the wavelet packet analysis (COIFMAN; WICKERHAUSER, 1992), which allows a more flexible decomposition guided by information theory. Works describing the CWRU data by wavelet packets is found in LIU (2012); WEI et al. (2011); LUO; YU; LIANG (2013); CHEBIL et al. (2009). An application to rotating machinery with an extended description on how to select the appropriate wavelet base is described in LIU (2005). However, it is not possible to modify the tree structure by optimization of the information contents of the leave nodes since it is necessary to obtain corresponding feature vectors for each sample. This means that a tree structure optimization for a normal machine condition could generate a wavelet packet tree that is different from the tree generated by a faulty condition thus not permitting the direct comparison of the features. Figure 16 shows two comparative examples extracted from real signals.

The wavelet package analysis is a time-frequency domain method which permits the level by level decomposition using a wavelet function. The decomposition results in 2^l signals, where l is the number of desired levels. The procedure proposed in (XIA et al., 2012) uses as mother wavelet Daubechies 4 and refining down to the fourth decomposition level. The energy calculated in the leaf nodes are used as final features. Only the leaf nodes were used to calculate the features of two bearings. The total number of wavelet package analysis features is $16 \times 2 = 32$.

3.1.2.3 Complex Envelope Spectrum

There are some frequency groups involved in a typical bearing fault. First there is the natural high natural frequency (resonance) of the ball when hitting the defective region which can be located on itself, the cage, interior or exterior raceway. Low frequencies are contributed mainly by shaft rotation related faults, like unbalance or misalignment. It is necessary to establish a model of the bearing to understand these frequency groups. The structure of a rolling bearing allows to establish a model of possible faults. The bearings, when defective, present characteristic frequencies depending on the localization of the defect (RAGULSKIS; YURKAUSKAS, 1989; MOBLEY, 1999; RIEGER; CROFOOT, 1977). There are four characteristic frequencies at which faults can occur. Knowing the shaft rotational frequency FS, the fault frequencies that can be calculated are the fundamental cage frequency FC, ball pass inner raceway frequency FBPI, ball pass outer raceway



(a) 0.007 inch, 0 hp, outer race fault signal decomposition

(b) 0.021 inch, 3 hp, outer race fault signal decomposition

Figure 16 – Reconstructed signal in wavelet packet tree of depth $j = 4$ for two outer race faults, varying with respect to the fault severity and work load. Only the first $0.1s$ of a single sample are processed and shown in order not to overburden the graph.

frequency FBPO and the ball spin frequency FB (MCINERNY; DAI, 2003). For the ball bearings with angular contact with the cage, the outer ring is static and the inner ring rotates at the shaft speed. Figure 17 illustrates a basic model of a bearing with the rolling elements, the inner and outer raceways and the cage.

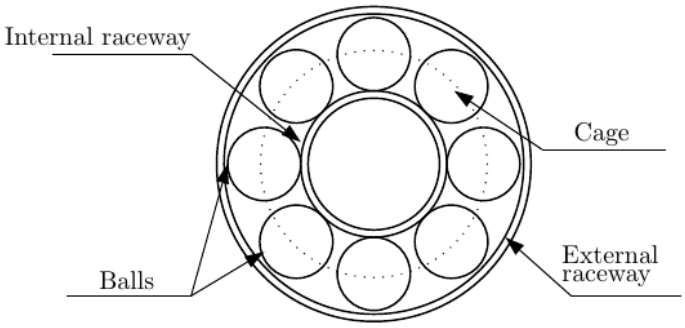


Figure 17 – Sectional view of a bearing model (MOBLEY, 1999).

The complex envelope analysis is illustrated in figure 18. First a high pass filter is applied in order to eliminate the influence of the low frequency vibrations caused by noise, unbalance and misalignment (figure 18c). Subsequently, an analytical signal is calculated by applying the Hilbert transform to the original signal and adding it in quadrature to it. The magnitude of the Fourier transform of the analytical signal translates the characteristic bearing faults frequencies to the low frequency band (figure 18d). The final features are the narrow band energy around the expected fault frequencies and their harmonics. Six harmonics were calculated for each of the two bearings considered.

This kind of feature extraction needs a specific feature for each fault, because it tries to identify high energy where the faults manifest themselves. This work intends to identify three types of faults, ball, inner race and outer race, using six harmonics for two bearings. Considering that each bearing produces features to identify failures in the other one, because they have different dimensions, the total number of complex envelope analysis features is $3 \times 6 \times 2 \times 2 = 72$. The final features are the narrow band energy around the expected fault frequencies and their harmonics.

3.1.3 Summary of the CWRU Datasets

Table 9a and table 9b show the summary of the two datasets generated from the CWRU vibratory data. The field division explains if the dataset has explicit divisions for training, validation and test. In the cases presented in tables 9a and 9b, none of the datasets have explicit division.

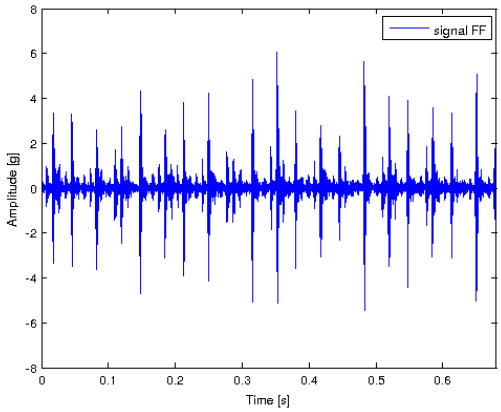
# Samples	2295
# Features	130
# Classes	19 (unbalanced)
Multiclass	Yes
Division	none

(a) CWRU bearing dataset.

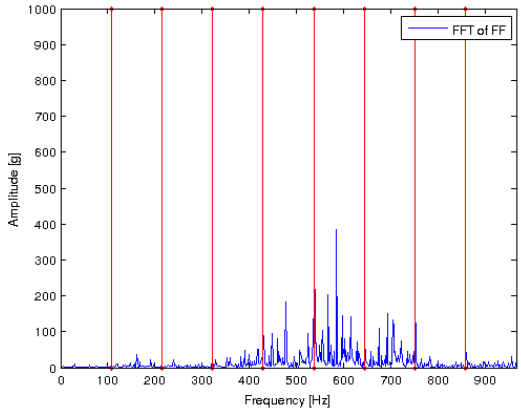
# Samples	600
# Features	130
# Classes	12 (balanced)
Multiclass	Yes
Division	none

(b) Special AUC-ROC CWRU bearing dataset.

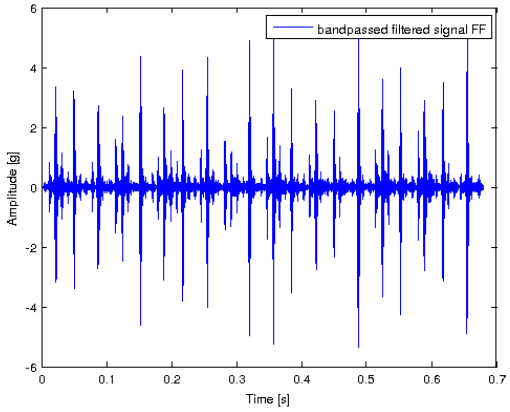
Table 9 – Summary of the datasets generated with the CWRU bearing data.



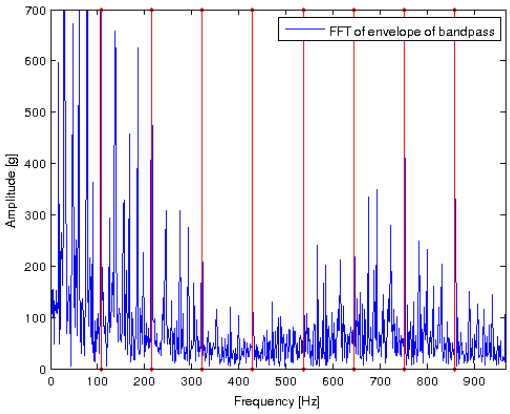
(a) Original signal in time domain.



(b) Frequency domain transformed from the original signal.



(c) Remaining high frequency components in time domain.



(d) Final frequency domain data.

Figure 18 – Complex Envelope Analysis Procedure Steps. The feature are extracted from the final frequency domain data.

3.2 Tennessee Eastman Chemical Process

In an effort to provide a realistic testbed for control and condition monitoring tasks in a chemical engineering context, a software simulator of a production plant was proposed in DOWNS; VOGEL (1993). A reactor, condenser, stripper, compressor and separator constitute the main components of the system where two liquid products and a liquid byproduct in two parallel reactions are obtained. The original code is written in Fortran, a Matlab/Simulink adaptation is also available (WASHINGTON, 2015). The usefulness of the Tennessee Eastman Chemical Process simulator is corroborated by recent publications from the field of fault diagnosis (BOLDT; RAUBER; VAREJÃO, 2017; GAO; HOU, 2016; XU; DENG, 2016; HE; GENG; ZHU, 2015; XU; CHEN; ZHU, 2014; BOLDT; RAUBER; VAREJÃO, 2014a; YIN; LUO; DING, 2014; YIN et al., 2012). In these references, the data produced by the simulator in CHIANG; BRAATZ; RUSSELL; MIT (2001, 2001) are used as input to the diagnosis system. Figure 19 outlines the schema

Table 10 – Manipulated variables of the production process (DOWNS; VOGEL, 1993)

Variable name	Acronym	Base case initial value (%)	Low limit	High limit	Units
D feed flow (stream 2)	XMV (1)	63.053	0	5811	kg h^{-1}
E feed flow (stream 3)	XMV (2)	53.980	0	8354	kg h^{-1}
A feed flow (stream 1)	XMV (3)	24.644	0	1.017	kscm h^{-1}
A and C feed flow (stream 4)	XMV (4)	61.302	0	15.25	kscm h^{-1}
Compressor recycle valve	XMV (5)	22.210	0	100	%
Purge valve (stream 9)	XMV (6)	40.064	0	100	%
Separator pot liquid flow	XMV (7)	38.100	0	65.71	m^3h^{-1}
Stripper liquid product flow	XMV (8)	46.534	0	49.10	m^3h^{-1}
Stripper steam valve	XMV (9)	47.446	0	100	%
Reactor cooling water flow	XMV (10)	41.106	0	227.1	m^3h^{-1}
Condenser cooling water flow	XMV (11)	18.114	0	272.6	m^3h^{-1}

of the Tennessee Eastman simulator. The Tennessee Eastman Chemical Process (TE) has eleven manipulated variables and 41 measured variables. Table 10 shows the eleven manipulated variables of the process and table 11 the 41 measured variables. All variables, manipulated and measured, are merged into the feature vector of dimension $11 + 41 = 52$ that reflects the complete description of the process state. The comparison of methods that use this industrial process is usually performed with the data provided by CHIANG; BRAATZ; RUSSELL (2001).For the normal process operation and the 21 faults, an explicit separation into training and test data is provided. The interval between the acquisition of two consecutive signal samples is 3 minutes. The normal class has 1460 samples (73 h) for training (file `d00.dat`) and 960 samples (48 h) for test (file `d00_te.dat`). The 21 faults have 480 samples (24 h) for training (files `d01.dat` to `d21.dat`) and 960 samples (48 h) for test (files `d01_te.dat` to `d21_te.dat`). However, the first 160 samples (8 h) of the fault test files correspond to normal operation before the fault is inserted into the simulator.

3.2.1 Identified Classes

The considered fault classes (besides the normal class) are listed in table 12. Originally 20 faults were defined in DOWNS; VOGEL (1993). An additional sticking valve fault was defined in CHIANG; BRAATZ; RUSSELL (2001). This work identifies the 21 faults in table 12 against the normal condition. Thus, there are actually 21 binary datasets, with explicit separation between training and test. Since the TE process has explicit separation of training and test datasets, it can be tested in a relatively fast way, because no cross-validation is needed. For non-parametric stable classifiers, as k-NN, repeated experiments are useless, because all rounds produce the same result. Then, for this kind of classification method, just one experiment round is necessary. On the other hand, unstable classifiers, as ANN, need several rounds. This kind of classification method has some

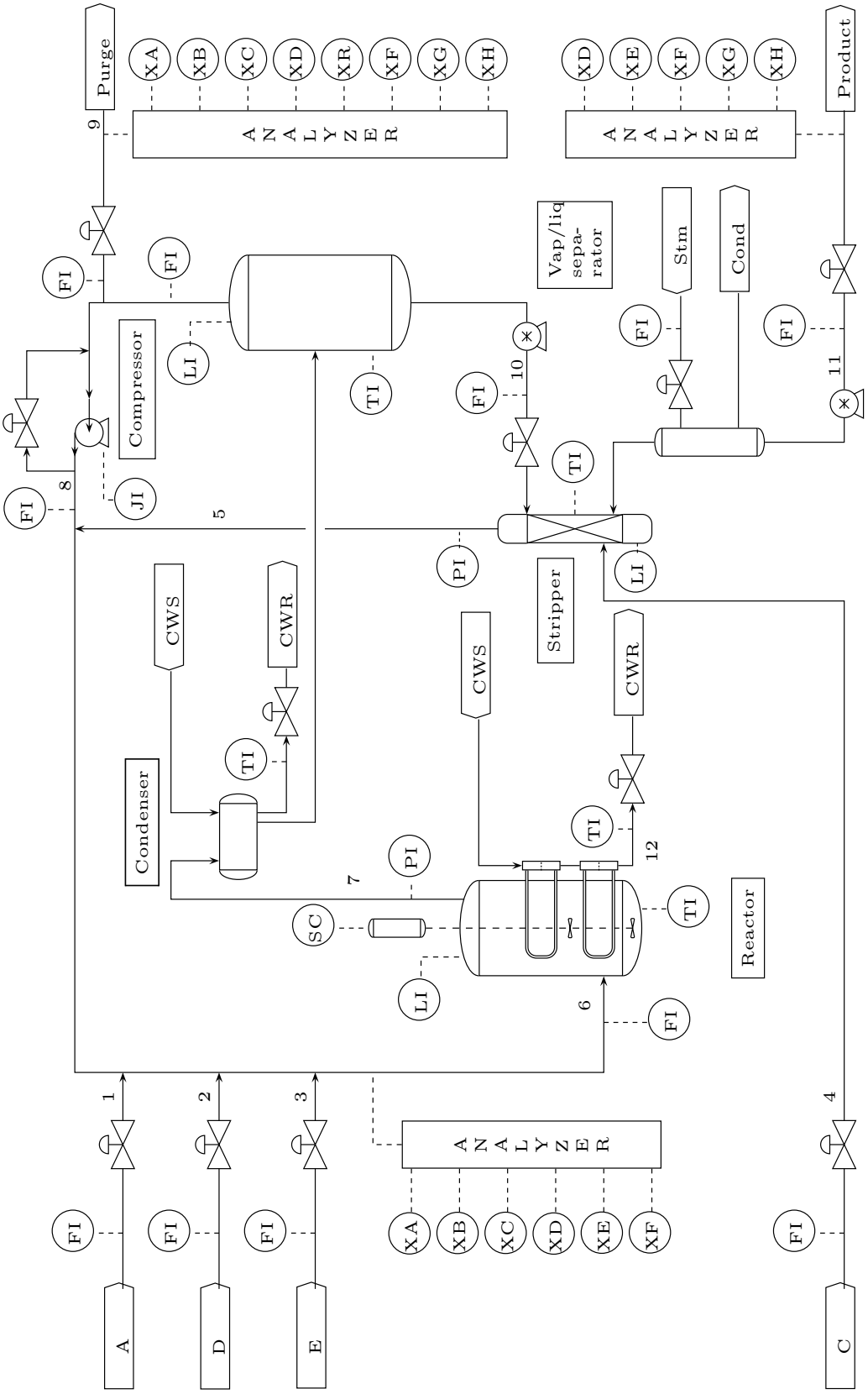


Figure 19 – The Tennessee Eastman chemical plant simulator (DOWNS; VOGEL, 1993).

randomness in its training that can produce different results for the same data division.

Table 11 – Process measurements (DOWNS; VOGEL, 1993)

Block name	Variable name	Variable number	Base case value	Units
Input feed	A feed (stream 1)	XMEAS (1)	0.25052	kscm h^{-1}
	D feed (stream 2)	XMEAS (2)	3664.0	kg h^{-1}
	E feed (stream 3)	XMEAS (3)	4509.3	kg h^{-1}
	A and C feed (stream 4)	XMEAS (4)	9.3477	kscm h^{-1}
Reactor	Reactor feed rate (stream 6)	XMEAS (6)	42.339	kscm h^{-1}
	Reactor pressure	XMEAS (7)	2705.0	kPa gauge
	Reactor level	XMEAS (8)	75.000	%
	Reactor temperature	XMEAS (9)	120.40	°C
Separator	Product separator temperature	XMEAS (11)	80.109	°C
	Product separator level	XMEAS (12)	50.000	%
	Product separator pressure	XMEAS (13)	2633.7	kPa gauge
	Product separator underflow	XMEAS (14)	25.160	m^3h^{-1}
Stripper	Stripper level	XMEAS (15)	50.000	%
	Stripper pressure	XMEAS (16)	3102.2	kPa gauge
	Stripper underflow	XMEAS (17)	22.949	m^3h^{-1}
	Stripper temperature	XMEAS (18)	65.731	°C
	Stripper steam flow	XMEAS (19)	230.31	kg h^{-1}
Miscellaneous	Recycle flow (stream 8)	XMEAS (5)	26.902	kscm h^{-1}
	Purge rate (stream 9)	XMEAS (10)	0.33712	kscm h^{-1}
	Compressor work	XMEAS (20)	341.43	kW
	Reactor cooling temperature	XMEAS (21)	94.599	°C
	Separator cooling temperature	XMEAS (22)	77.297	°C
Reactor feed analysis	Component A	XMEAS (23)	32.188	mol%
	Component B	XMEAS (24)	8.8933	mol%
	Component C	XMEAS (25)	26.383	mol%
	Component D	XMEAS (26)	6.8820	mol%
	Component E	XMEAS (27)	18.776	mol%
	Component F	XMEAS (28)	1.6567	mol%
Purge gas analysis	Component A	XMEAS (29)	32.958	mol%
	Component B	XMEAS (30)	13.823	mol%
	Component C	XMEAS (31)	23.978	mol%
	Component D	XMEAS (32)	1.2565	mol%
	Component E	XMEAS (33)	18.579	mol%
	Component F	XMEAS (34)	2.2633	mol%
	Component G	XMEAS (35)	4.8436	mol%
	Component H	XMEAS (36)	2.2986	mol%
Product analysis	Component D	XMEAS (37)	0.01787	mol%
	Component E	XMEAS (38)	0.83570	mol%
	Component F	XMEAS (39)	0.09858	mol%
	Component G	XMEAS (40)	53.724	mol%
	Component H	XMEAS (41)	43.828	mol%

Table 12 – Process faults (DOWNS; VOGEL, 1993). Each fault has its own training and test datasets. Each fault training dataset is merged with the normal condition training dataset, since the former has only fault samples. The test datasets have 160 samples of normal condition and 800 of its respective fault condition.

Variable number	Process variable	Type
IDV (1)	A/C feed ratio, B composition constant (stream 4)	Step
IDV (2)	B composition, A/C ratio constant (stream 4)	Step
IDV (3)	D feed temperature (stream 2)	Step
IDV (4)	Reactor cooling water inlet temperature	Step
IDV (5)	Condenser cooling water inlet temperature	Step
IDV (6)	A feed loss (stream 1)	Step
IDV (7)	C header pressure – reduced availability (stream 4)	Step
IDV (8)	A, B, C feed composition (stream 4)	Random variation
IDV (9)	D feed temperature (stream 2)	Random variation
IDV (10)	C feed temperature (stream 4)	Random variation
IDV (11)	Reactor cooling water inlet temperature	Random variation
IDV (12)	Condenser cooling water inlet temperature	Random variation
IDV (13)	Reaction kinetics	Slow drift
IDV (14)	Reactor cooling water valve	Sticking
IDV (15)	Condenser cooling water valve	Sticking
IDV (16)	Unknown	Unknown
IDV (17)	Unknown	Unknown
IDV (18)	Unknown	Unknown
IDV (19)	Unknown	Unknown
IDV (20)	Unknown	Unknown
IDV (21)	Valve fixed at steady state	Sticking

3.2.2 Summary of TE Chemical Process Datasets

Table 13 shows the summary of the datasets generated using the TE chemical process simulator. It is important to highlight that each condition has two datasets. One for training and one for test. Since the training datasets of the failures have only defective examples (480 samples), each one is merged with the training dataset for the normal condition (1460 samples). The defective test datasets have normal and defective condition (960 samples). Therefore, they are not merged with the normal test dataset. The datasets are used only for binary classification.

# Samples	1460+480=1940 for training and 960 for test
# Features	52
# Classes	22 (unbalanced)
# Datasets	21 for training and 21 for test
Multiclass	no
Division	training & test

Table 13 – Summary of TE chemical process dataset.

3.3 Electrical Submersible Pumps

In some cases, oil wells elevate its product naturally to the surface. When an increase in the fluid pressure or in the production rate is needed, an artificial support method has to be applied. Electric Submersible Pump (ESP) systems are often used as an artificial lifting method in offshore oil exploration. ESP systems utilizes a submerged multistage centrifugal pump driven by an electrical motor, whose power is supplied from the surface by an electric cable (TAKÁCS, 2009). These systems work inside the oil well and its installation and eventual removal due to maintenance are expensive operations. Before an ESP system is put into operation in sub-sea installations, it needs to be carefully tested for avoiding initial failures, and high intervention costs. Thus, to avoid posterior problems during the operational phase, rigorous reliability evaluation is performed before ESP deployment (TAVNER et al., 2008). This evaluation is made in laboratory where large amounts of data are used by experts who analyze the ESP system. In vibratory analysis, accelerometers are attached to several ESP body positions, during an operational test in which water is pumped. In the test studied in this work, 36 accelerometers are fixed pairwise with a 90 degrees phase offset in the axial direction (BOLDT et al., 2014). The accelerometers are equally distributed along the motors, protectors and pumps. Hence, one pair of accelerometers is connected at the bottom, middle and top of each component, as shown in figure 20. Finally, the collected data is analyzed and labeled by an expert, using as the principal technique the visual inspection of the frequency spectra obtained from the Fourier transform of the raw vibration signal.

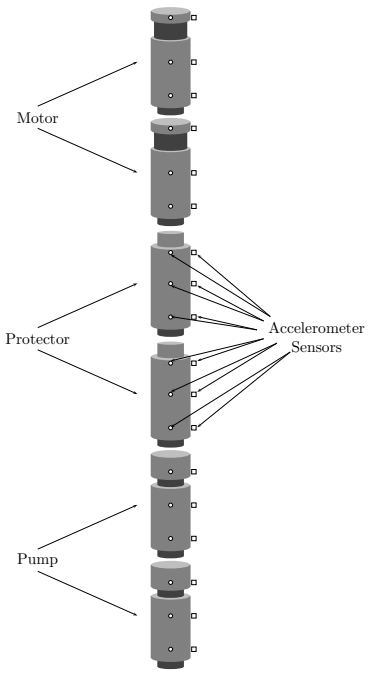


Figure 20 – Accelerometers position to test an ESP system. Six accelerometers are placed in each ESP component.

Human experts usually expend several days to provide the final diagnostic. Not rarely, the decision whether the pump system is able to be put into operation relies on professional experience (BOLDT; RAUBER; VAREJÃO, 2014a). An automatic diagnosis system can help human experts, experienced or novice, to identify failures. Supervised learning is often used to develop model-free diagnosis systems (WANDEKOKEM et al., 2011). The model-free approach has the advantage of avoiding explicit expert knowledge. Its main drawback is the necessity of a significant amount of labeled examples. In the work presented in OLIVEIRA-SANTOS et al. (2016), a comparative study of classification methods applied to diagnose faults in ESPs is presented. That work uses eight features extracted from the frequency domain and presents the K-Nearest-Neighbour as an equivalent classifier to Random Forest, Support Vector Machine and Decision Trees, when the training data is standardized.

The vibration data is collected with a sample rate of 4096 Hz by accelerometers placed in the ESP system, as shown in figure 20. When a human expert performs a diagnosis, the data in the time domain is transformed to the frequency domain and plotted to allow a better visual inspection, as shown in figure 21. For each graph, the large dashed red line, on the superior part (between amplitude 0.1 and 0.2), and the dashed dotted yellow line (near to amplitude 0.3), are similar to alarm thresholds where the amplitude usually cannot be higher. However, they are actually guidelines, because there are situations where the amplitude surpasses them and the fault is not considered, and situations where the amplitude does not surpass them but the fault is considered. The diagnostic also depends on the signature of the other components, or the component behavior in lower and higher frequency rotations. Thus, the decision if there is a fault or not frequently relies on experience of the human expert.

3.3.1 Identified Classes

This study tries to create an expert system that does not depend on a threshold or considers different signals to diagnose a component. Consequently, additional features have to be added to enhance the classification performance. Figure 21 shows the characteristic signature of each condition considered in OLIVEIRA-SANTOS et al. (2016) and in this study. The conditions can have very different graphical manifestations, but those in figure 21 can be considered the expected behavior. All five charts were plotted with real data. The dataset used in this work has 4570 examples distributed a priori as shown in table 14.

Typical normal condition is presented in figure 21e, where all amplitudes are below the large dashed red line and the dashed dotted yellow line. Figure 21a shows the unbalance signature, that is characterized by a high peak in the $1x$ frequency, where x is the rotation frequency of the shaft. A misalignment fault signature is characterized by an abnormal high peak in the $2x$ frequency. It can be seen in 21b, even though the amplitude does

Table 14 – A-priori percentages of normal and fault classes.

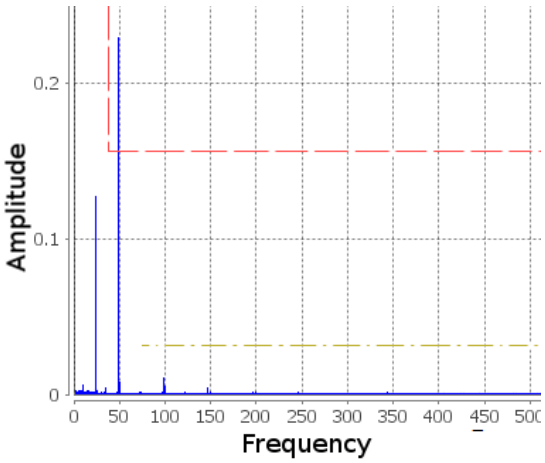
Condition class	A priori class distribution [%]
Normal condition	81.10
Unbalance	10.61
Misalignment	1.09
Rubbing	0.77
Accelerometer fault	6.43

not surpass the dashed dotted yellow guide line, the misalignment fault was considered. Rubbing is characterized by high energy in the lower frequencies and the presence of peaks in $0.5x$, $1x$ and $2x$. It is represented by the example of figure 21c. It is not uncommon that accelerometers fail. Despite it is not an ESP defect, it caches the eventual existence of an ESP fault. Thus, the accelerometer fault is also considered. Its signature presented in figure 21d. It has high energy in the lower frequencies, but does not present any peak in the $0.5x$, $1x$ and $2x$ frequencies.

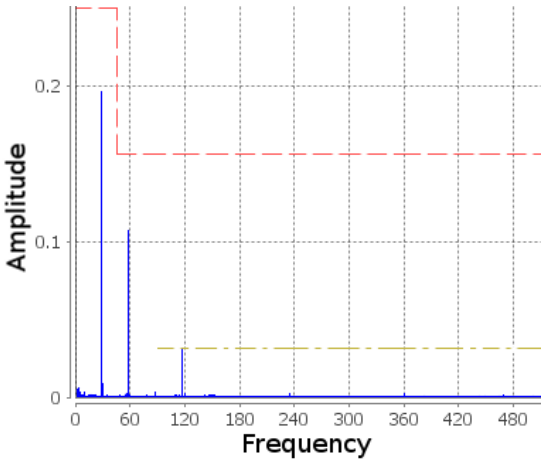
Since the size of each time domain signal is much higher than the number of examples, this signal cannot be used directly in supervised learning (RAUBER; BOLDT; VAREJÃO, 2015). Thus, descriptive features must be extracted before training a classifier via supervised learning. The features extracted in OLIVEIRA-SANTOS et al. (2016) are described below:

- x_{rf}: Rotation frequency (first harmonic) of the submersible motor pump during the test;
- x_{rfm}: Magnitude in the rotation frequency (first harmonic);
- x_{rfm2}: Magnitude in the double of the rotation frequency (second harmonic);
- x_{rfrms}: Root mean square of the magnitudes around the rotation frequency, [x_{rf} - 1, x_{rf} + 1];
- x_{rfmm}: Median of the magnitudes around the rotation frequency, [x_{rf} - 1, x_{rf} + 1];
- x_{m3to5}: Median of the magnitudes of the low frequencies, interval [3Hz, 5Hz];
- x_{ilr}: Intercept (a) of the linear regression of logarithm of the frequency magnitudes (Mag) over the interval of frequencies (Fq) [5Hz, 19Hz], equation 3.1;
- x_{slr}: Slope (b) of the linear regression of logarithm of the frequency magnitudes (Mag) over the interval of frequencies (Fq) [5Hz, 19Hz], equation 3.1.

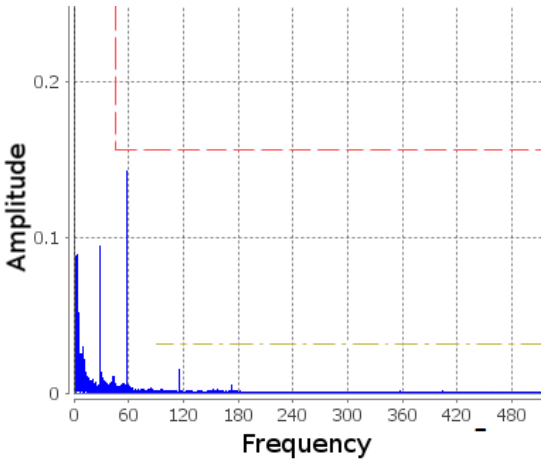
$$\log(Mag) = a - b \times Fq \tag{3.1}$$



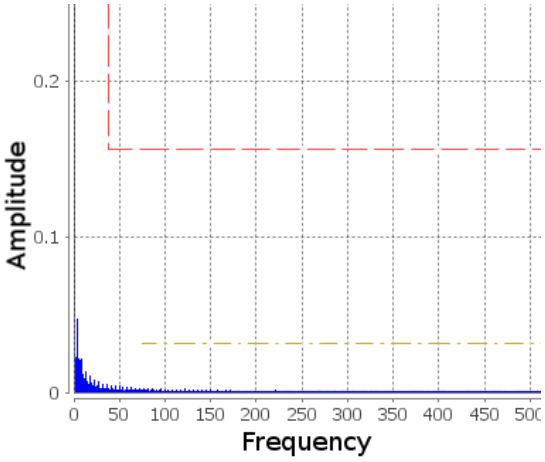
(a) Unbalance. High peak in $1x$.



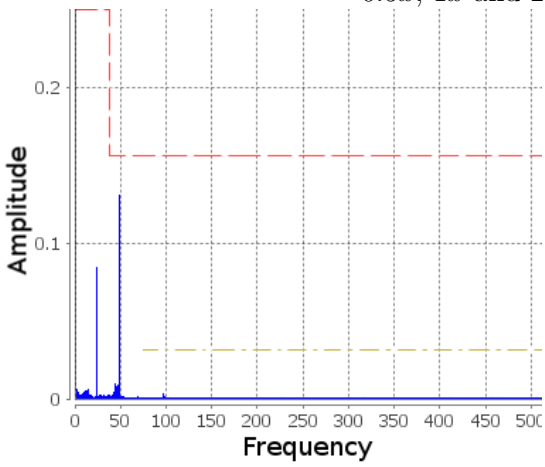
(b) Misalignment. High peak in $2x$.



(c) Rubbing. High energy before $0.5x$ with presence of peaks in $0.5x$, $1x$ and $2x$.



(d) Accelerometer Fault. High energy before $0.5x$ with absence of peaks in $0.5x$, $1x$ and $2x$.



(e) Normal Condition. All frequencies below the threshold lines.

Figure 21 – Typical frequency signatures for the five considered categories. The dashed dotted yellow line and the large dashed red line are thresholds indicating normal operational behaviour of general frequencies and of frequencies with peaks respectively.

Following the principle of heterogeneous feature extraction presented in RAUBER; BOLDT; VAREJÃO (2015), twelve more features have been added to the dataset used

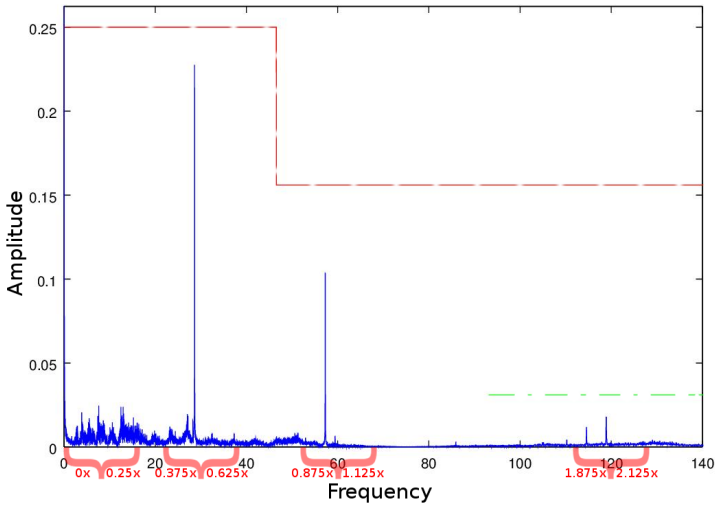


Figure 22 – Frequency bands of the additional feature set. The large dashed red line and the dashed dotted green line are similar to thresholds which the amplitude usually cannot surpass them. The red curly brackets show the frequency windows from where the additional features were extracted.

in OLIVEIRA-SANTOS et al. (2016). The additional features are the average, standard deviation and the maximum of four frequency windows. A zoom-in of the spectra in figure 22 shows the considered intervals by red curly brackets. The intervals are $0x$ to $0.25x$, $0.375x$ to $0.625x$, $0.875x$ to $1.125x$ and $1.875x$ to $2.125x$. The example in figure 22 was collected from an ESP system working in a frequency rotation of 60 Hz ($1x = 60$). It is worth to highlight that 60 Hz is the nominal frequency rotation, that is the frequency desired for a given power supply. However, the natural friction of the components always reduce the real rotation. That is why the high peak ($1x$) in figure 22 is presented in a smaller frequency than 60 Hz. The difference of nominal and real rotation frequency is the main motivation to use frequency windows instead the direct amplitude from the desired frequency.

3.3.2 Summary of ESP Dataset

Table 15 shows the summary of the ESP dataset. This dataset has no explicit division for training, validation and test. Therefore, cross-validation is required to use it.

# Samples	4570
# Features	20
# Classes	5 (unbalanced)
Multiclass	Yes
Division	none

Table 15 – Summary of ESP dataset.

Part II

Proposals and Results

4 Heterogeneous Feature Models

The main hypothesis of this chapter is that higher discriminative power of automatic fault detection systems can be achieved when as much information as possible is extracted from the raw signal and the final number of features is reduced to a reasonable amount by subsequent feature selection. This means that various feature extraction paradigms can be combined in parallel to create a quite heterogeneous pool of candidate features from which the final set is obtained by feature selection. RAUBER; BOLDT; VAREJÃO (2015) is the first publication that proposes a fusion of features that stem from completely different signal feature extraction methods. After this, other publications have used heterogeneous feature models to diagnose in several domains (BROETTO; VAREJÃO, 2016; BOLDT et al., 2017). The experimental results suggest that this strategy of pooling, followed by selection in general improves the performance of the fault diagnosis. Fig. 23 illustrates the general framework to perform rotating machinery fault diagnosis. The sequence of information processing steps is:

1. *Signal feature extraction*: Use the available sensors attached to the machine to acquire the raw signal in the time domain. From the raw signal, considering a feature model, extract the feature vector within this model. Do this for all available models.

2. *Feature pooling*: Assemble a global feature vector which contains as much information as possible about the machine condition. Reduce the number of features by recombination of the existing ones (extraction) or by retaining only a subset (selection), c.f. the next two steps.

3. *Dimensionality reduction*: It can be done exclusively or sequentially by feature extraction on the feature level and feature selection.

    a) *Feature extraction on the feature level*: From the existing feature vector extract new features. Usually the dimension is reduced considerably. The new features are abstract descriptions of the machine condition. The most prominent linear method is Principal Component Analysis (PCA).

    b) *Feature selection*: Either discard or preserve existing features to form a subset of the existing features. The main goal is dimensionality reduction and the increase of discriminatory power of the net feature set.

4. *Classification*: Define a single or multiple classifier model and estimate its performance.

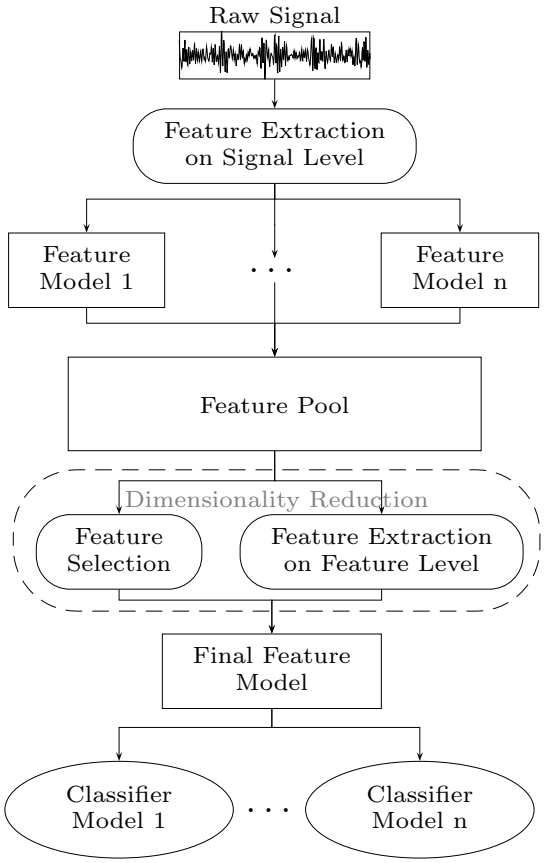


Figure 23 – Computational intelligence framework for bearing fault diagnosis.

The main contribution of RAUBER; BOLDT; VAREJÃO (2015) is the simultaneous use of distinct types of features and the posterior phase of feature selection. This strategy ensures that no important information is omitted during the extraction phase and that no irrelevant information from the extraction phase is preserved. The next sections present the experimental results of experiments using the feature models presented in section 3.1 with the CWRU data.

4.1 Performance without Feature Selection

Table 16 presents the estimated performance of the 1-NN, SVM and MLP classifiers with datasets extracted by three different feature models and the merged pool of all features, joining the three models. All 130 features were used to train the classifier and run the test, i.e. no feature selection was performed. The 1-Nearest Neighbor classifier always uses leave-one-out to estimate the performance parameters. For the SVM and MLP classifiers this method is computationally too expensive, therefore a 10-fold cross validation was employed. In order to improve the robustness of the estimation the mean over 10 different of such 10-fold runs was taken. The values in bold face highlights the best classifier for each feature model, while the underlined values indicates the best model for each classifier.

Table 16 – Performance (accuracy and AUC-ROC) of three classifiers using datasets with different feature extraction models, where the Complete Pool is the union of those other three models. Values in bold face highlights the best classifier for each feature model. Underlined values indicates the best model for each classifier.

	ACC			AUC-ROC		
Feature model	1-NN	SVM	MLP	1-NN	SVM	MLP
Statistical	96.25%	**97.25%**	93.90%	0.9239	**0.9976**	0.9827
Wavelet Package	**99.83%**	99.63%	99.04%	**1.0000**	0.9999	0.9915
Complex Envelope Spectrum	**97.65%**	96.88%	93.63%	0.9581	0.9998	**0.9999**
Complete Pool	99.96%	98.13%	**99.97%**	0.9917	0.9991	**1.0000**

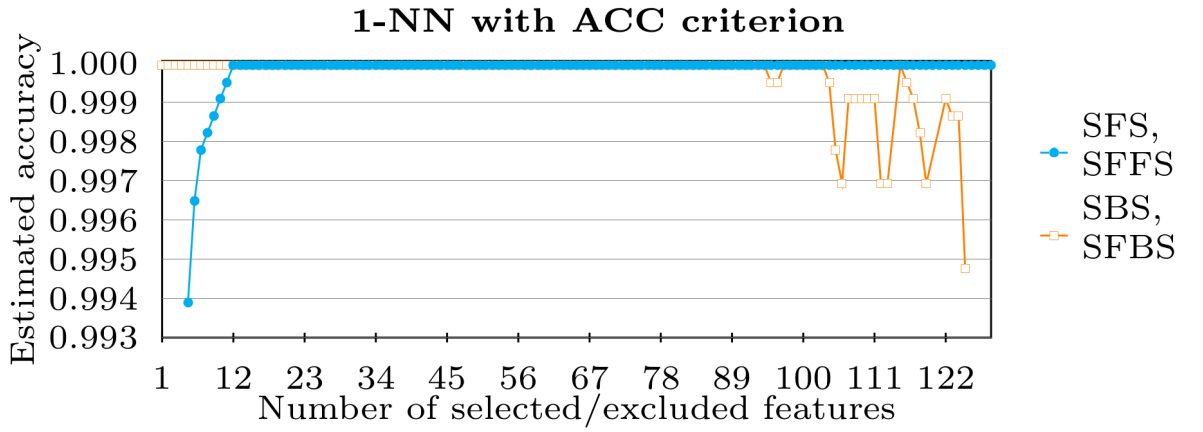
The results of table 16 give a valuable feedback considering the classification task and the feature models. The accuracy (ACC) and AUC-ROC suggest that the CWRU data set is easy to distinguish with respect to the defined classes. The statistical feature model seems to be the least discriminative. The wavelet package energy is by itself the best feature model. The complete pool is not always the best feature model. An analysis of the results provides a principal motivation of the need of the posterior feature selection. When pooling all feature models, the classification results degrade, since some features contain more noise than information. The subsequent feature selection can improve performance and simultaneously diminish the complexity of the feature model.

4.2 Performance with Feature Selection

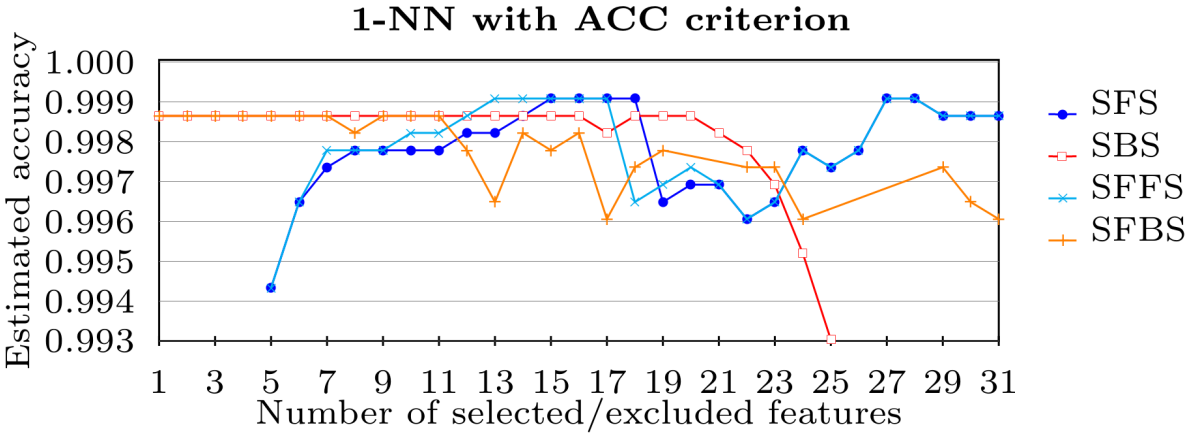
The results of the first experiment using feature selection performed in this work is presented in figure 24. This figure shows the evolution performance of four feature selection methods with the 1-NN classifier. In both graphs, for the Sequential Forward Selection (SFS) and Sequential Floating Forward Search (SFFS) algorithms, the horizontal axis represents the number of features selected. For the Sequential Backward Selection (SBS) and Sequential Floating Backward Search (SFBS) algorithms, the horizontal axis represents the number of features excluded.

Fig. 24a and fig. 24b shows the estimated accuracy of the 1-NN classifier using leave-one-out cross-validation during the process of feature selection. The algorithm uses the wrapper approach with the performance value as the proper selection criterion, the estimated accuracy in this case. All three feature models are tested, plus the global pool. Only the wavelet package graph is shown, the statistical and envelope features exhibit a similar behavior, but with smaller performance. For the global pool, SFFS and SFBS have identical behaviour as SFS and SBS respectively. In general, the floating techniques seem to perform better than the sequential search in the case of the wavelet package model. The

selected subset of all feature models together enters a error free saturation in a very early stage. The analysis of this result emphasizes the hypothesis that selected features from the global feature pool have superior performance compared to the individual feature models.



(a) Estimated accuracy during feature selection for all features.



(b) Estimated accuracy during feature selection for the wavelet package model.

Figure 24 – Evolution of the feature selection algorithms with 1-NN algorithm and leave-one-out validation.

4.3 Comparison using AUC-ROC as Performance

For accuracy estimation, all classes are considered separately. However, the classifiers with feature selection easily reach 100% of accuracy for the CWRU dataset. Therefore, for AUC-ROC estimation all classes of table 17 are merged, except the second one in the table which is considered as the positive class (Ball_FE_1_007). This fault of the ball at fan end was identified as the hardest to be distinguished in preliminary tests. Figure 25 compares the AUC-ROC estimated using the pool of features and the best single feature extraction model, i.e. wavelet package. In both graphs, 25a and 25b, the yellow curve is the highest and shows the best AUC-ROC performance acquired for each dataset.

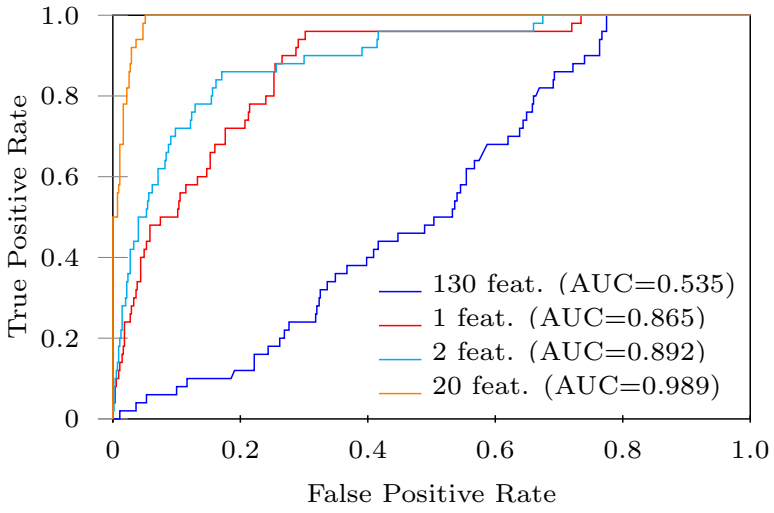
Table 17 – Classes of the special AUC-ROC dataset, varying fault severity and load for the fan end ball fault. The bold face fault (Ball_FE_1_007) was chosen as the positive class, since it was identified as the most difficult to classify in prelininary experiments.

Class	Name	Samples	Distribution	Description
1	Ball_FE_0_007	50	8.33%	0.007", 0 hp load
2	**Ball_FE_1_007**	**50**	**8.33%**	**0.007", 1 hp load**
3	Ball_FE_2_007	50	8.33%	0.007", 2 hp load
4	Ball_FE_3_007	50	8.33%	0.007", 3 hp load
5	Ball_FE_0_014	50	8.33%	0.014", 0 hp load
6	Ball_FE_1_014	50	8.33%	0.014", 1 hp load
7	Ball_FE_2_014	50	8.33%	0.014", 2 hp load
8	Ball_FE_3_014	50	8.33%	0.014", 3 hp load
9	Ball_FE_0_021	50	8.33%	0.021", 0 hp load
10	Ball_FE_1_021	50	8.33%	0.021", 1 hp load
11	Ball_FE_2_021	50	8.33%	0.021", 2 hp load
12	Ball_FE_3_021	50	8.33%	0.021", 3 hp load

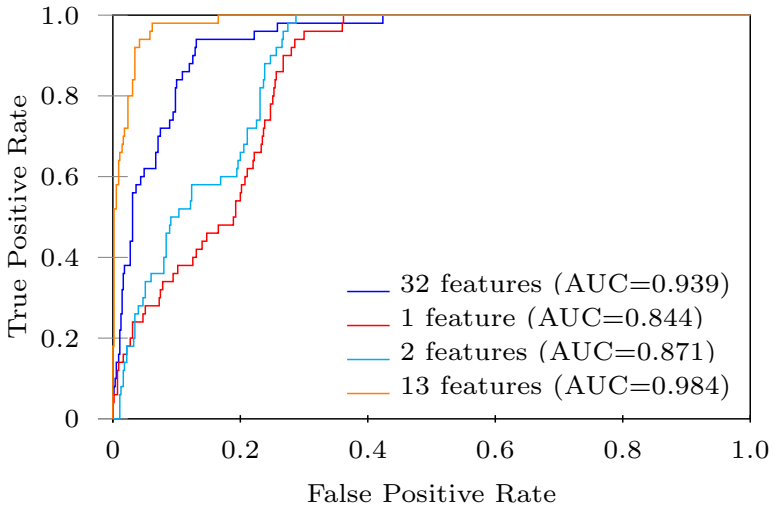
The pool of features presented a smaller AUC-ROC performance than the wavalet package model when all features are used. On the other hand, when the features are selected, the AUC-ROC value estimated with the pool of features is higher. This result illustrates the importance of feature selection over the pool of features.

4.4 Comparison of Feature Selection and Feature Extraction by PCA

Paying tribute to a consolidated dimensionality reduction technique in fault diagnosis, see e.g. YIN et al. (2012) and VONG; WONG; IP (2013), a comparison of feature *selection* to feature *extraction*, on the feature level, is done, represented by its most prominent linear method, Principal Component Analysis (PCA). The basic difference is that PCA produces new features as a recombination of the existing ones, whereas feature selection leaves the original features unchanged. The new PCA features are ranked by their variance. Since PCA is an unsupervised method, the maximization of variance does not necessarily augment class discernability. Figure 26 shows a comparison of AUC-ROC estimated using the pool of features and the wavelet package model. The dataset used to perform this experiment is that presented in table 17. Figure 26 compares the ROC curves for the SVM with the wavelet dataset and the dataset with the pool of features. The red and cyan lines shows the classifier performance for small amount of PCA components, i.e. 1 and 2. The blue lines shows the ROC for all non-zero PCA components. For instance, the original amount of features in the dataset used in figure 26a is 130. However, after the PCA process, three components became null (zero). The yellow line shows the ROC with the best AUC for each feature model.



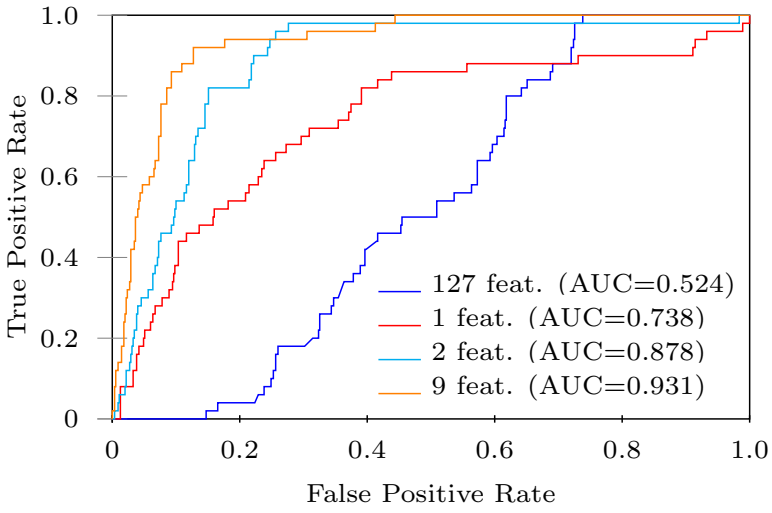
(a) AUC-ROC estimated for SVM and SFS from all features.



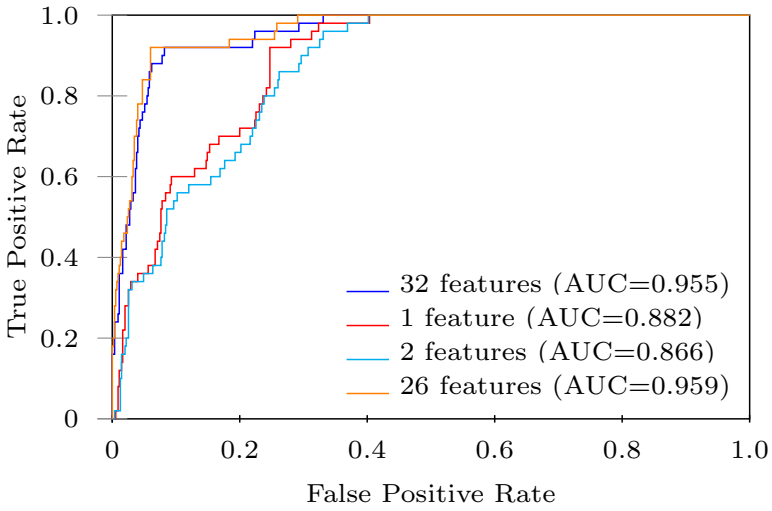
(b) AUC-ROC estimated for SVM and SFS from features of the wavelet package model.

Figure 25 – Comparison of AUC-ROC estimated for SVM with SFS from all features and the wavelet package model.

Using PCA instead feature selection, wavelet package had better performance than the pool of features. Then, heterogeneous feature models worked better with feature selection then PCA for this dataset, also using the dataset presented in table 17. Therefore, table 18 compares the performance criteria for the KNN and SVM classifiers, comparing selection (SFS) and extraction (PCA). SFS was chosen because all feature selection methods presented in figure 24 has similar behavior, but SFS was the fastest one. The conditions presented in table 17 were considered. The highest estimated value for each combination of classifier, dimensionality reduction method and feature model is presented in table 18. Feature selection achieved higher performance in all datasets and measures. It can also be observed that the pool of features achieved higher classification performance than all other models, for both, accuracy and AUC-ROC, even when PCA was used.



(a) AUC-ROC estimated for SVM and PCA from all features.



(b) AUC-ROC estimated for SVM with PCA from features of the wavelet package model.

Figure 26 – Comparison of AUC-ROC estimated for SVM and PCA from all features and the wavelet package model.

4.5 Comparion of Feature Selection Methods Using G-mean as Performance

Two feature selection methods were tested with 10 rounds of 10-fold paired nested cross-validation. The complete dataset with 19 classes and the pool of features was used to perform this experiment. Since G-mean is also an indicated metric for unbalanced datasets, it was also used here. However, unlike the comparison made in the previous section, these experiments did not choose a class for being the positive class. Instead of it, the final performance is the average of the g-mean of each class. Table 19 shows only a small improvement given by the feature selection methods. Considering the used dataset, the feature selection methods behavior presented in table 19 is theoretically expected,

Table 18 – Ball FE fault classification performance. Criterion shown together with the best result found with feature selection and PCA.

		Complex Envelope	Pool	Statistical Features	Wavelet Package
		Accuracy			
KNN	SFS	36: **0.553**	54: **0.758**	06: **0.485**	21: **0.668**
	PCA	65: 0.470	96: 0.652	13: 0.300	16: 0.610
SVM	SFS	32: **0.554**	12: **0.792**	09: **0.573**	13: **0.758**
	PCA	16: 0.499	24: 0.721	06: 0.333	15: 0.663
		AUC-ROC of second class of table 17			
KNN	SFS	35: **0.786**	54: **0.941**	08: **0.718**	13: **0.888**
	PCA	69: 0.655	06: 0.744	04: 0.640	13: 0.795
SVM	SFS	25: **0.954**	24: **0.991**	05: **0.877**	16: **0.981**
	PCA	03: 0.866	38: 0.975	06: 0.836	26: 0.970

since the classification performance for the the classifier without feature selection is nearly to the maximum possible. When feature selection methods are used as wrapper and the classification metric is high after the selection or removal of few features, the classification performance in the inner-validation tends to reach its maximum value. Then, the feature selection method considers all remaining features to be equally good, turning the selection after this point merely aleatory. A possible way to overcome this behavior is choosing another metric that is more rigorous than G-mean. In this context, a rigorous measure would be one that decrease its values considerably with few wrong classifications. Next section presents experiments with another metric indicated for unbalanced datasets that might be more rigorous than G-mean.

Table 19 – Comparison using G-mean.

	KNN	**KNN-RANK**	**KNN-SFS**
Min.	**0.9681**	0.9431	0.9431
1st Qu.	0.9864	0.9864	**0.9867**
Median	0.9915	**0.9924**	0.9915
Mean	0.9913	0.9913	**0.9916**
3rd Qu.	0.9976	0.9976	**0.9979**
Max.	**1.0000**	**1.0000**	**1.0000**

4.6 Selecting Features Using F-Measure as Performance

Another comparison between the pool of features and the wavelet package was made using the F1-measure macro-averaged as performance. F1-measure is also indicated for unbalanced datasets. This experiment was executed with 10 rounds of 10-fold paired

nested cross-validation and the complete dataset with 19 classes and the pool of features. Figure 27 compares the performance the 1-NN with the wavelet package dataset (KNN-W) versus the 1-NN with the pool of features (KNN-P), the 1-NN with the ranking feature selection and the wavelet dataset (KNN-RW) versus the 1-NN with the ranking feature selection and the pool of features (KNN-RP) and the random forest algorithm with the wavelet package dataset (RF-W) versus the random forest algorithm with the pool of features (RF-P). It can be seen that the pool of features presents a small improvement for the 1-NN classifier. On the other hand, the classification performance of the random forest algorithm is worse with the pool of features.

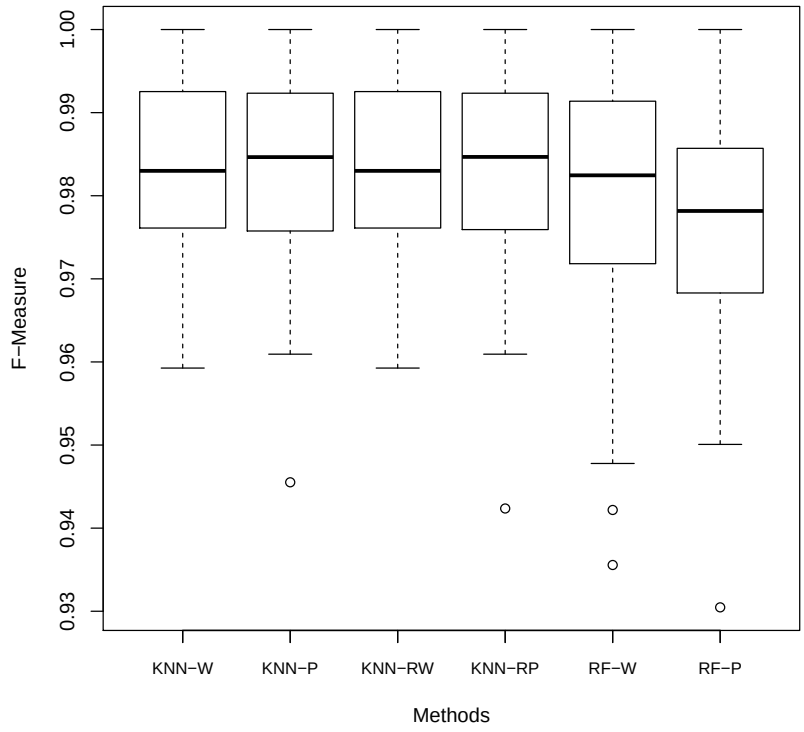


Figure 27 – F-Measure boxplot with CWRU datasets. KNN indicates the 1-NN classifier and RF, random forest. W means Wavelet package extraction model dataset and P means pool of features. R represents the ranking feature selection method.

Figure 28 shows that despite the higher performance presented by the KNN-RP methods, there is no statistical significant difference among the classification methods. Figure 29, 30, 31 and 32 shows the individual performance of each class with the six classification methods experimented. The worse general performance was for the normal class, with all methods.

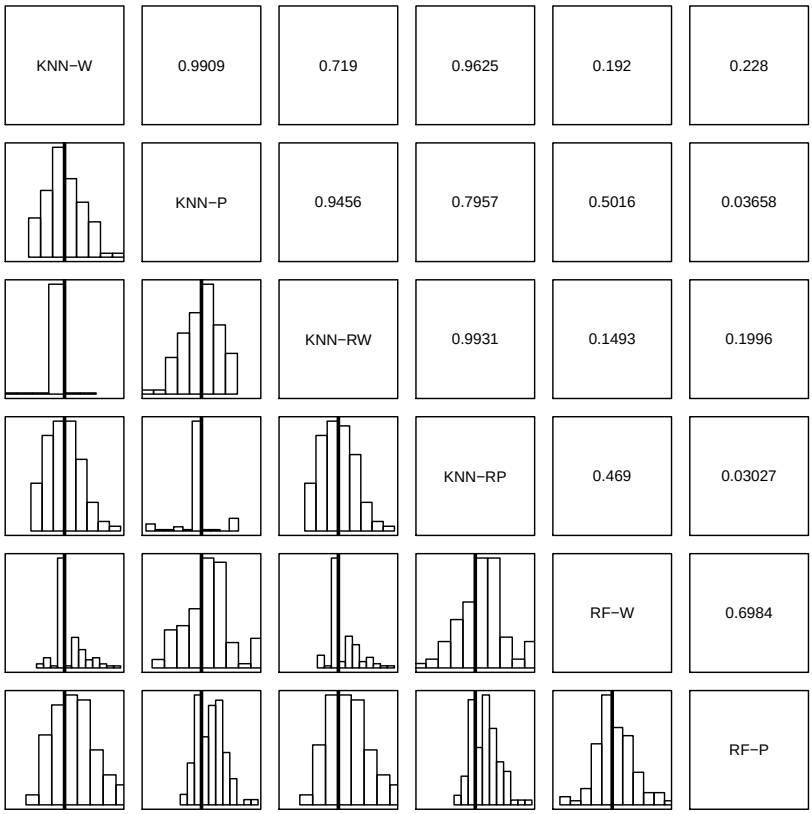


Figure 28 – Pairwise comparison between methods. The lower triangle shows the histograms of the differences of the Fscores for each pair of method. The upper triangle presents the corresponding p-values of the correlated t-test.

Chapter Considerations

The process description is taken to a higher abstraction level by characterizing the process conditions by multiple heterogeneous feature models. The essence is that it really does not matter where the the descriptive information comes from. As long as new information can be obtained from a feature model, its contribution is welcome. This approach by itself would have a low impact on the discriminative power of the diagnosis system. Only if a posterior information filter in the form of feature selection is used, the general performance can be expected to improve. A considerable amount of publications tries to compensate a poor feature model by an over-sophisticated classifier model. A different approach is taken here. If a good process description is available, a simple classifier is sufficient. In this case the description appears in the form of a selection of many available features, stemming from quite distinct sources. The experiments suggest that this approach is a promising methodology, extensible to other areas of industrial applications.

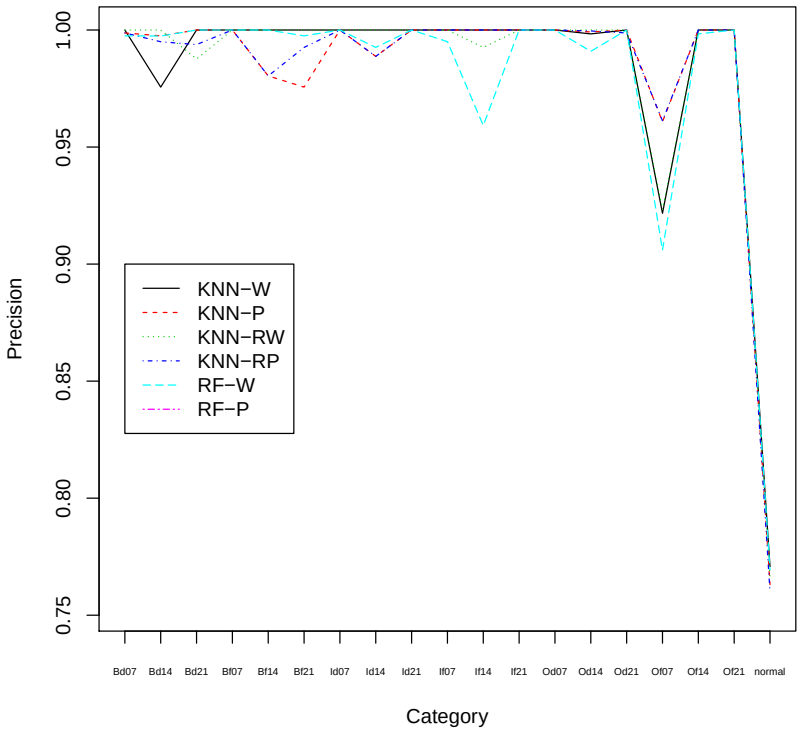


Figure 29 – Precision per condition.

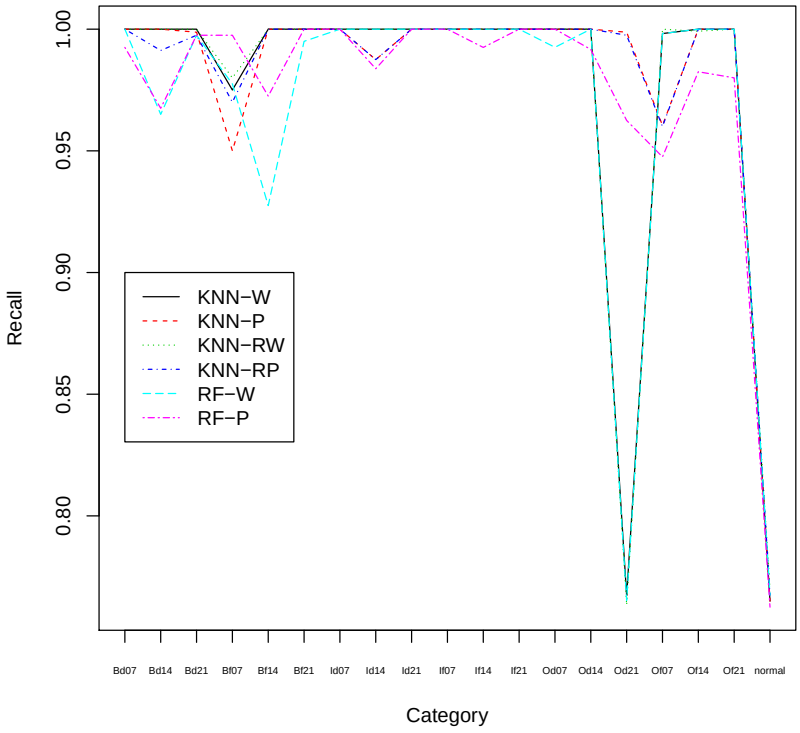


Figure 30 – Recall per condition.

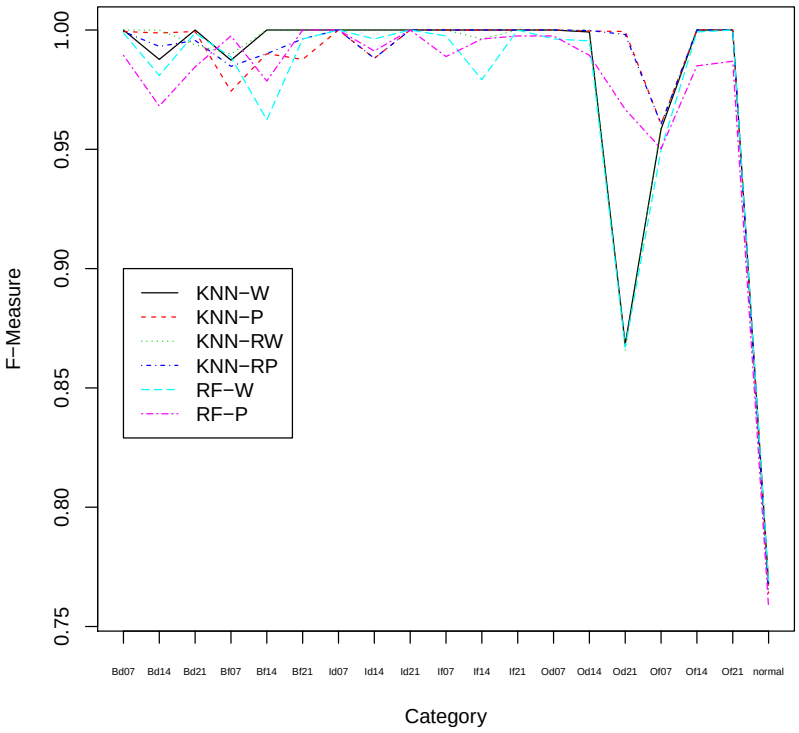


Figure 31 – F-Measure per condition.

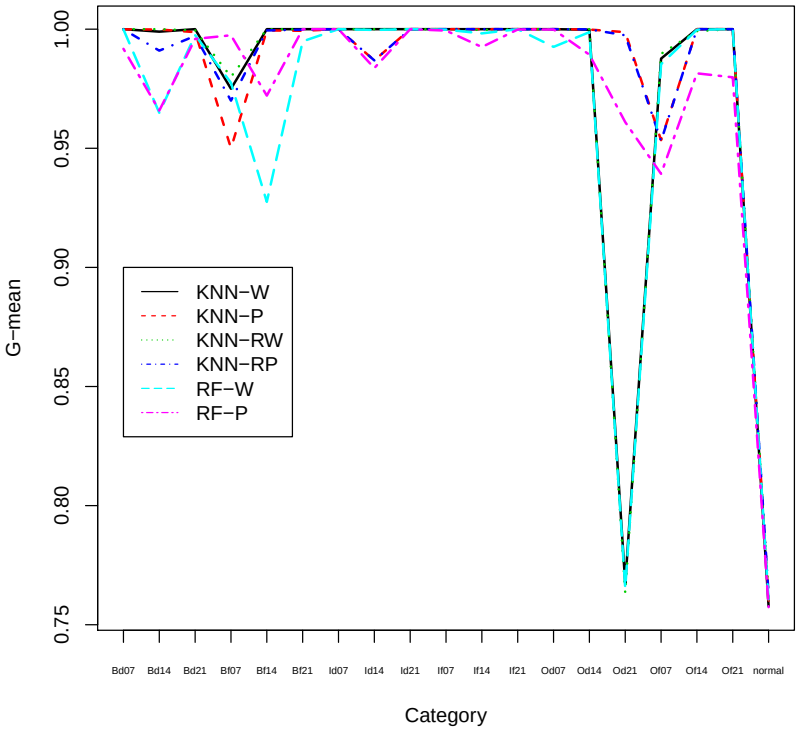


Figure 32 – G-Mean per condition.

5 Cascade Feature Selection

Inspired by the idea of Cascade Classifiers for face recognition (VIOLA; JONES, 2001), it is presented the concept of Cascade Feature Selection (CFS) (BOLDT; RAUBER; VAREJÃO, 2017). Cascade Classifiers is a method that combines increasingly more complex classifiers in a "cascade". It discards background regions of an image quickly while spending more computation on promising object-like regions. Since regions with low probability to have an object are initially discarded by fast and weak classifiers, the strong algorithms have to cope with a reduced set of regions and are consequently faster than when they were used alone. The concept of CFS follows a similar idea. Weak and fast feature selection methods, like ranking with a filter approach, places the top of the cascade to filter possible "bad" features. Strong and slow feature selection methods, like GA as wrapper, act at the bottom of the cascade, where less features remain and a more precise analysis is needed.

Despite the inspirational role that the Cascade Classifiers plays for CFS, these two approaches are quite different, in a way that CFS is an original contribution. Thus, it is worth to highlight some differences between these two approaches. The first major difference is that Cascade Classifiers is only applicable to detect objects in images, while CFS is able to select features in different contexts of supervised or unsupervised learning. Cascade Classifiers returns a trained ensemble of classifiers. This ensemble is ready to classify new samples. CFS returns a subset of features that can be used to train a classifier and then classify new samples. Therefore, the result of CFS can be used in diverse scenarios and with any classifier model. The classifiers can be either ensembles or single classifiers, that are lighter and faster than ensembles. Moreover, it allows a better understanding of the data set features and how their combinations influence in classification performance. How Cascade Classifiers and CFS work is also different. The Cascade Classifiers approach train a classifier ensemble, using different feature subsets to train the weak classifier of each cascade stage. The weak classifier term in VIOLA; JONES (2001) refers to a single classifier that compounds a cascade stage. CFS can use or not a classifier in each stage, since each stage forwards a subset of features to the next, instead of a trained classifier. Another important difference is that Cascade Classifiers uses each stage to predict the label of a new sample. CFS uses its stages only as intermediate methods to accelerate the feature selection process. Having been trained, a classifier does not need the CFS algorithm and its stages anymore.

CFS can achieve better results only if the algorithms at the top of the cascade filter possible bad features. However, many feature selection algorithms require the final number of features as a parameter. This requirement is not a deterrent problem, but a bad choice of the number of features might not lead to good feature subsets. Trying to

find the optimal number of features for each auxiliary feature selection, this work used the so called hybrid approach (BOLDT et al., 2015). In PENG; LONG; DING (2005) a SFS algorithm with a filter criterion (mRMR) is used in its first phase to generate a multivariate rank of the features. The second phase selects the optimal number of features with a wrapper approach. All possible feature subset sizes are tested and the best is chosen. In case of ties, the smaller subset is chosen. In this work the same strategy is applied for ranking (univariate) and SFS (multivariate) algorithms. Thus, these algorithms are here called Hybrid Ranking and Hybrid SFS, respectively. It is important to remark that the Hybrid Ranking is not a univariated method, since it only sorts the features by their individual evaluation, but the subset of features is chosen by evaluating combinations of features. The filter criterion used here is based on the Euclidean Distance among the samples, for the sake of speed performance. The GA based feature selection is not applied as a hybrid method to evaluate the TE data, only as wrapper. The F1-score (SOKOLOVA; LAPALME, 2009) average of three holdout 50/50 validations with the ELM classifier was used as wrapper criterion. This criterion was employed by the GA based feature selection and the wrapper parts of the hybrid algorithms.

Three CFS configurations were tested in this work:

1. Hybrid Ranking followed by GA (Rank-GA);
2. Hybrid SFS followed by GA (SFS-GA);
3. Hybrid Ranking followed by Hybrid SFS and then GA (R-S-GA).

Figure 33 shows an example of a two levels CFS using Hybrid Ranking and Genetic Algorithm. The feature sets are represented by right angle rectangles and the processes are represented by round corner rectangles. The CFS process, represented by a dotted round corner rectangle, receives the initial feature set and returns the reduced feature set. The first level of this CFS configuration is a Hybrid Ranking, represented by a dashed round corners rectangle. It evaluates each feature in the initial set $\mathcal{F}$ according to the filter criterion. As ranking is a univariate algorithm, it does not consider possible mutual dependence among the features. The result of the filter phase is a rank $\mathcal{R}$, where $|\mathcal{R}| = |\mathcal{F}|$. Then, a sequential algorithm tests all possible feature set sizes $n' \in \{1, \ldots, |\mathcal{R}|\}$ using a wrapper approach. The subset with the highest performance is chosen as the intermediate feature set $\mathcal{F}'$, where $\mathcal{F}' \subset \mathcal{F}$. It is important to remember that while a pure ranking algorithm is a univariate method, the hybrid ranking algorithm has a multivariate phase that evaluates all $|\mathcal{R}|$ possible subsets to choose the number of features n' for the intermediate subset $\mathcal{F}'$. Then, $\mathcal{F}'$ is delivered to the next algorithm in the cascade. The GA receives the intermediate feature set $\mathcal{F}'$ and delivers the final subset of features $\mathcal{G}$, where $\mathcal{G} \subset \mathcal{F}'$.

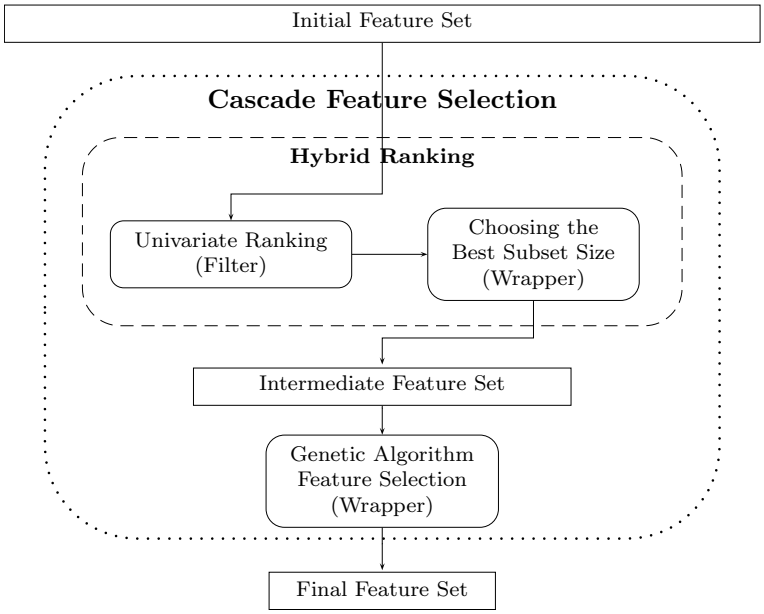


Figure 33 – Cascade composed by Hybrid Ranking and Genetic Algorithm.

Figure 34 shows the CFS configuration with the Hybrid SFS algorithm in the first level, represented by a dashed round corners rectangle. As SFS is a multivariate algorithm, the criterion evaluation is done using feature subsets instead of individual features. However, to implement this hybrid algorithm, SFS evaluates all features $f_i \in \mathcal{F}$ and returns the order in which each feature was selected, as done in PENG; LONG; DING (2005). In other words, the SFS sets $n \leftarrow |\mathcal{F}|$ and returns a rank $\mathcal{R}$, where $|\mathcal{R}| = |\mathcal{F}|$. This rank $\mathcal{R}$ is also used by the sequential algorithm to test all possible feature set sizes $n' \in \{1, \ldots, |\mathcal{F}|\}$, using a wrapper approach. Then, the subset with the best performance is chosen as the intermediate feature set $\mathcal{F}'$, where $\mathcal{F}' \subset \mathcal{F}$. There are some similarities

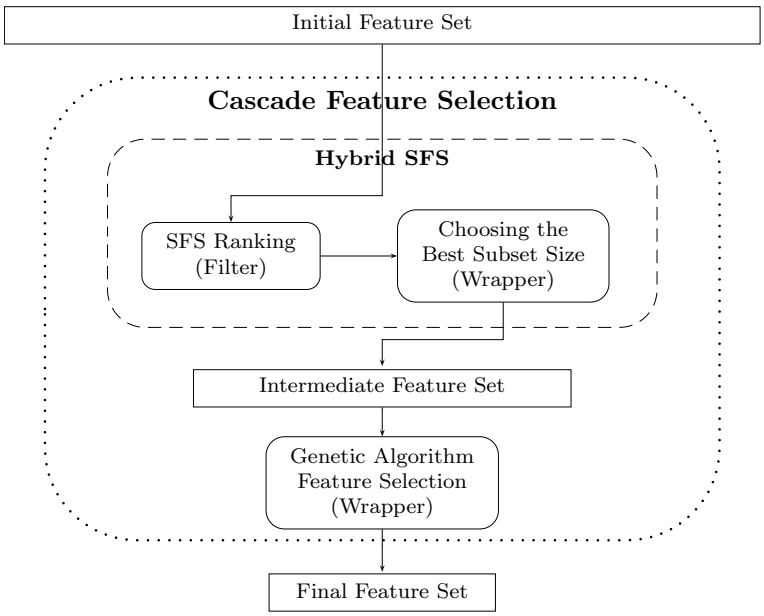


Figure 34 – Cascade composed by Hybrid SFS and Genetic Algorithm.

and differences between the hybrid ranking in figure 33 and the hybrid SFS in figure 34 that are worth to be highlighted. Both algorithms are divided into two parts. The first part is a ranking and the last part is a method to define the number of the features to be selected. However, unlike the hybrid ranking algorithm, the hybrid SFS algorithm uses a multivariate ranking in its first part. The multivariate ranking allows the hybrid SFS to investigate more feature combinations than the hybrid ranking. As the previous CFS configuration, a GA is the last stage of the cascade. It receives the intermediate feature set $\mathcal{F}'$ and delivers the final subset of features $\mathcal{G}$, where $\mathcal{G} \subset \mathcal{F}'$.

Figure 35 shows the most complex CFS configuration proposal in this work, that is a combination of the other two presented before. It has the fastest and weaker method (hybrid ranking) on the top of the cascade receiving the initial feature set $\mathcal{F}$. The hybrid ranking algorithm returns the first intermediate feature set $\mathcal{F}'$, where $|\mathcal{F}'| \leq |\mathcal{F}|$, performing the same process explained for figure 33. The hybrid SFS algorithm receives the first intermediate subset $\mathcal{F}'$ and returns the last intermediate feature set $\mathcal{F}''$, where $|\mathcal{F}''| \leq |\mathcal{F}'|$. The process performed by the hybrid SFS algorithm is the same as explained for figure 34. The GA is again the last stage of the cascade. It receives the last intermediate feature set $\mathcal{F}''$ and delivers the final subset of features $\mathcal{G}$, where $\mathcal{G} \subset \mathcal{F}''$.

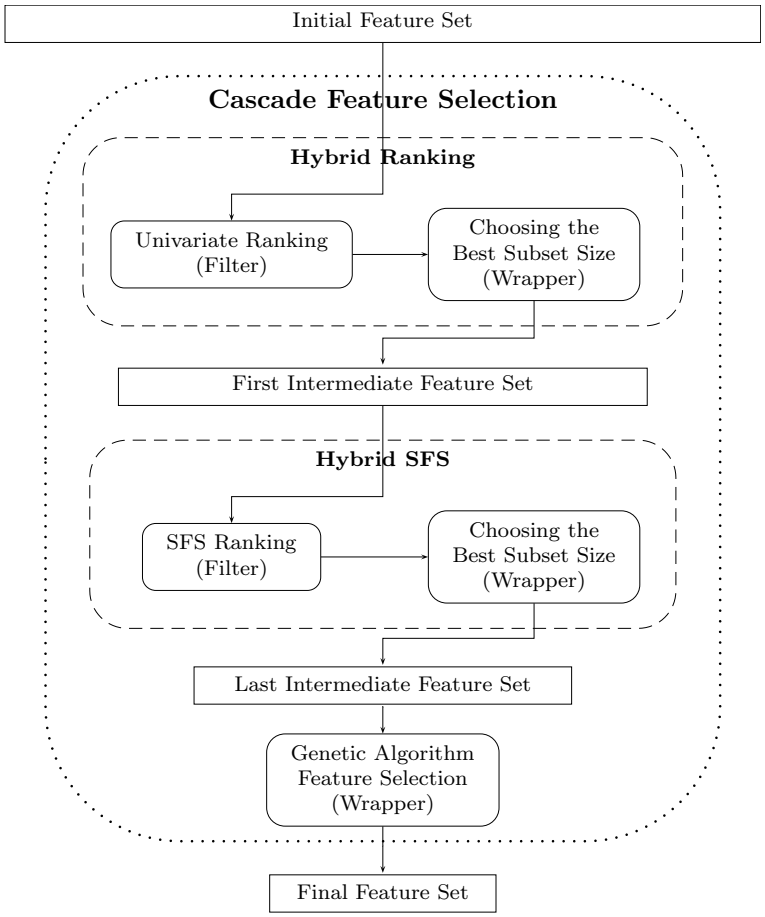


Figure 35 – Cascade composed by Hybrid Ranking, Hybrid SFS and Genetic Algorithm.

The CFS configurations presented in this section follow the CFS principles. As the GA is slower than Ranking and SFS, it could not be placed before them in the cascade. Moreover, preliminaries experiments showed that SFS was slower than Rank for the data sets used here. Then, SFS also could not be placed before the Ranking algorithm.

Two experimental approaches to judge the usefulness of the CFS proposal. The first evaluation, presented in section 5.1, is a comparison of CFS classification performance with the experimental results presented in GAO; HOU (2016). It is a recent paper that uses the TE data, provides a method with feature set dimensionality reduction and uses a sophisticated classifier (SVM). The same experimental procedure described in GAO; HOU (2016) was performed to generate the results of subsection 5.1, which uses the normal and four faulty process conditions in table 20 to train the classifier. The tests are performed using only the condition IDV01 of table 20. The second evaluation, presented in section 5.2, compares the three CFS configurations (Rank-GA, SFS-GA and Rank-SFS-GA) with the GA Feature Selection. All 21 abnormal conditions of the TE data were tested. Section 5.3 analyses the behavior of the methods taking into account the subset of features selected to generate the results presented in subsection 5.2. It is worth remembering that all experiments performed in this work are bias aware. Thus, under no circumstances during the training stage, neither the feature selection methods nor the classifier had access to the test data. Even the average and the standard deviation to scale the data sets were obtained exclusively from the training data sets. The precaution avoids any overoptimistic performance scores and allows a fair comparison of methods.

5.1 Comparison with GS-PCA

The method called GS-PCA uses Principal Component Analysis (PCA) to reduce the feature set dimension. Then, to increase prediction accuracy and reduce computational load, the optimization of SVM hyperparameters is accomplished by a grid search (GS). Unfortunately, the authors of GAO; HOU (2016) did not provide the range of the hyperparameters used in the GS stage. To replicate the experiments without these parameters is non-trivial. The values of GS-PCA in table 21 are those reported in GAO; HOU (2016). The evaluation of GS-PCA was done using the training data in table 20 (`d00.dat`, `d01.dat`, `d02.dat`, `d04.dat` and `d05.dat`), but only fault IDV01 (`d01_te.dat`) was used for testing.

Table 21 shows the comparison of GS-PCA with the GA based feature selection used as a pure wrapper method and three CFS configurations, i.e. Rank-GA, SFS-GA and Rank-SFS-GA. The algorithms GA, Rank-GA, SFS-GA and Rank-SFS-GA executed for ten rounds of training and test. It must be emphasized that the TE data has explicit separation of training and test data. Therefore, no re-sampling or cross-validation is needed

Table 20 – Descriptions of the operating conditions used in (GAO; HOU, 2016).

Variable	Affected process variable in case of a fault
IDV00	Normal
IDV01	A/C feed ratio, B composition constant
IDV02	B composition, A/C ratio constant
IDV04	Reactor cooling water inlet temperature
IDV05	Condenser cooling water inlet temperature

to produce replicable and comparable experiments. The average accuracy and its standard deviation are presented for each feature selection method. The information for the number of rounds or standard deviation is not provided by GAO; HOU (2016). Therefore, this information is not shown for GS-PCA in table 21. It is important to remark that the running time presented in table 21 is the time expended to test the trained classification model and it does not include the training time. Training time comparison is made in table 22 and subsection 5.2. The results in table 21 emphasize the superiority of the four feature selection methods over GS-PCA.

Table 21 – Comparison with (GAO; HOU, 2016) considering only IDV01 as test dataset.

	Accuracy (%)	Running time (s)
GS-PCA	96.77	1.35
GA	99.50 (0.10)	5.70E-2 (6.75E-3)
Rank-GA	99.54 (0.12)	4.70E-2 (1.06E-2)
SFS-GA	99.67 (0.12)	4.30E-2 (6.75E-3)
R-S-GA	99.53 (0.12)	4.40E-2 (5.16E-3)

Since the main objective of CFS is to speed up the feature selection process, table 22 illustrates the comparison among the methods in terms of the number of selected features and the training time. Table 22 points out that all CFS configurations expended less time to select the features than the GA method. SFS-GA and R-S-GA selected less features than GA, while Rank-GA selected 0.2 features more than GA, in average. In the next subsection, another experiment design evaluates the methods based on feature selection, where the three CFS configurations are compared to the GA feature selection.

Table 22 – Average Number of Features and Average Training Time for each Feature Selection Method.

	Number of Features	Training Time (s)
GA	27.6 (2.2)	1.52E+4 (3.41E+3)
Rank-GA	27.8 (3.0)	1.41E+4 (2.40E+3)
SFS-GA	24.7 (2.3)	1.26E+4 (2.16E+3)
R-S-GA	25.6 (3.3)	1.38E+4 (4.41E+3)

5.2 Comparison with Genetic Algorithm

This section aims at comparing the CFS proposal to the feature selection method based on Genetic Algorithm, which is a widely known and admittedly good algorithm for feature selection. Unlike the procedure used in GAO; HOU (2016), the evaluations performed for this subsection verify the performance of each feature selection algorithm for all 21 abnormal conditions presented in MIT (2001). For each abnormal condition and each feature selection method, a complete feature selection was performed over a data set composed by the training data file of normal condition `d00.dat` and the training data file of the abnormal condition to be evaluated. This process is illustrated in figure 36.

The phases of training and test are represented by dashed round corner rectangles. It can be seen that the information flows only from the training phase to the test phase. Never the other way around. The right angle rectangles represent data and the round corner rectangles represent processes. Considering the fault xy, where $xy \in \{01, \ldots, 21\}$, the training process joins the normal training data set (`d00.dat`) with the xy training data set (`d`xy`.dat`) to create the initial training data set $\mathcal{X}$. The standardization process transforms the data set $\mathcal{X}$ into the standardized data set $\mathcal{X}' \leftarrow \frac{\mathcal{X}-\mu}{\sigma}$. The average μ and the standard deviation σ are kept for further standardization of the test data. The features

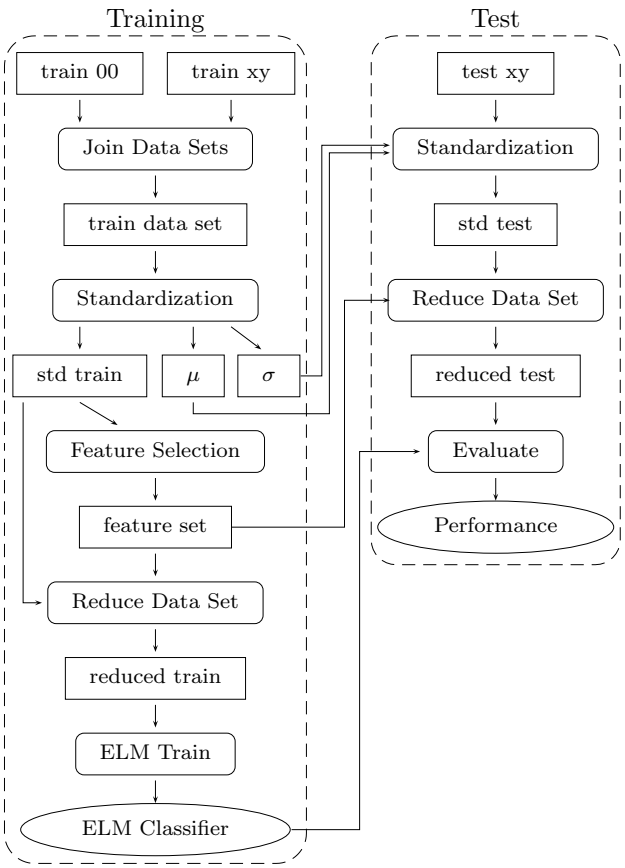


Figure 36 – Evaluation process performed to compare GA and CFS.

are selected from the data set $\mathcal{X}'$. The feature selection algorithms might use a classifier model or not, whether a wrapper feature selection is performed or not. All experiments performed used at least one wrapper stage, and consequently, an ELM model to select the features. In order to perform the wrapper evaluations without using the test data sets, the F1-score average of three holdout 50/50 validations were used over the data set $\mathcal{X}'$. By virtue of the data set $\mathcal{X}'$ and the feature set returned by the feature selection algorithm, a reduced training data set $\mathcal{X}''$ is generated. This set serves as the input for the ELM parameter learning, that produces the trained ELM classifier. In the test phase, the test data set $\mathcal{X}_{test}$ for fault xy (`d`xy`_te.dat`) is standardized using the average μ and the standard deviation σ calculated by the standardization process of the training phase ($\mathcal{X}'_{test} \leftarrow \frac{\mathcal{X}_{test}-\mu}{\sigma}$). The standardized data set $\mathcal{X}'_{test}$ is then reduced, keeping only the selected features. Finally, the performance scores are obtained using the reduced data set $\mathcal{X}''_{test}$ and the ELM classifier.

The first comparison evaluates the accuracy of each method. Table 23 shows the accuracy and standard deviation of each method for each condition. The line 'Mean'

Table 23 – Accuracy of each method for each condition.

	GA	Rank-GA	SFS-GA	R-S-GA
d01	*99.61 (0.14)*	**99.72 (0.05)**	**99.77 (0.04)**	99.74 (0.09)
d02	*96.66 (0.99)*	96.20 (0.65)	**97.95 (0.76)**	97.35 (0.89)
d03	50.16 (3.67)	50.01 (3.83)	47.65 (1.97)	52.42 (2.46)
d04	99.97 (0.05)	100.0 (0.00)	99.95 (0.05)	100.0 (0.00)
d05	**99.82 (0.07)**	*70.45 (0.34)*	99.82 (0.09)	*72.51 (1.22)*
d06	96.35 (6.72)	99.85 (0.10)	99.79 (0.00)	99.79 (0.00)
d07	*99.95 (0.05)*	99.97 (0.05)	**100.0 (0.00)**	99.92 (0.04)
d08	*73.57 (7.72)*	**83.98 (1.21)**	74.94 (4.92)	68.56 (6.85)
d09	*46.90 (3.41)*	**53.31 (0.30)**	**60.94 (4.83)**	**52.48 (0.48)**
d10	*66.48 (2.67)*	**69.32 (1.26)**	**70.69 (2.06)**	66.85 (2.90)
d11	*81.11 (2.29)*	80.94 (0.00)	**83.25 (0.40)**	**82.81 (0.22)**
d12	***88.25 (2.65)***	**90.21 (0.00)**	88.69 (2.06)	*85.88 (1.63)*
d13	*69.62 (7.78)*	66.67 (0.00)	**76.67 (3.51)**	**80.98 (0.97)**
d14	99.90 (0.00)	99.90 (0.00)	99.90 (0.00)	99.90 (0.00)
d15	*52.55 (3.08)*	50.73 (0.00)	**56.37 (0.18)**	**56.56 (1.54)**
d16	*74.64 (3.81)*	**78.02 (0.00)**	74.23 (1.01)	**82.17 (5.53)**
d17	93.90 (6.33)	94.79 (0.00)	92.94 (0.92)	96.08 (0.35)
d18	***81.93 (6.63)***	**89.79 (0.00)**	**89.96 (0.75)**	*41.23 (24.55)*
d19	**87.91 (2.48)**	*82.92 (0.00)*	87.52 (0.40)	*82.42 (0.04)*
d20	82.78 (6.34)	83.23 (0.00)	83.15 (0.18)	85.38 (0.09)
d21	92.43 (12.05)	96.25 (0.00)	88.06 (17.26)	96.25 (0.00)
Mean	***82.59 (3.76)***	**82.68 (0.37)**	**84.39 (1.97)**	*80.92 (2.37)*
Better	4	7	9	5
Worse	12	2	0	4

presents the average of each method across the conditions, and between parenthesis is the average of the standard deviation across the conditions. For the CFS methods, values in bold face are considered statistically significantly better than GA, according to a Student's t-test with a confidence level of $\alpha = 0.05$. Values in italic are significantly worse than GA. For GA, values in bold face are considered significantly better than at least one CFS method, and values in italic are significantly worse than at least one CFS method. Therefore, it might occur situations when GA is significantly better than one CFS method and also significantly worse than other, e.g. condition d12. For these cases, the values are in bold and italic font. When there is no significant difference among the methods, all values are in normal font, e.g. conditions d03 and d04. The lines 'Better' and 'Worse' counts how many conditions a method has values with bold or italic face, respectively. The significance test was also executed for the methods average. It can be seen that the Rank-GA and the SFS-GA configurations are significantly better than GA, in average. On the other hand, R-S-GA is, in average, significantly worse than GA, even though being significantly better for five conditions and significantly worse for four conditions. Moreover, there are conditions that the R-S-GA had the best accuracy among all tested methods.

Table 24 – Number of features selected by each method for each condition.

	GA	Rank-GA	SFS-GA	R-S-GA
d01	*24 (2.8)*	**15.4 (6.4)**	**14.6 (4)**	**9.7 (0.5)**
d02	*17.8 (1.3)*	**7.5 (0.5)**	**11.8 (1.5)**	**9.6 (3.2)**
d03	*11 (0)*	11.1 (0.3)	**9.5 (1.1)**	**9.9 (0.9)**
d04	*9.7 (2.6)*	**1 (0)**	**1 (0)**	**1 (0)**
d05	*18 (2.4)*	**8.2 (0.6)**	**4.9 (1.4)**	**6.8 (0.4)**
d06	*20.8 (3)*	**1 (0)**	**1 (0)**	**1 (0)**
d07	*12.6 (3.6)*	**1.9 (1.4)**	**1.4 (0.8)**	**3.4 (1.3)**
d08	*11.9 (1.3)*	**5 (0)**	10 (2.1)	**7.6 (0.8)**
d09	*11 (0)*	**6 (0)**	**5.2 (0.4)**	**5.4 (0.8)**
d10	*6.7 (0.5)*	**5.3 (0.5)**	**5.4 (0.8)**	**5.4 (0.8)**
d11	2.2 (0.6)	2 (0)	2 (0)	2 (0)
d12	*10.4 (1.1)*	**7 (0)**	**9 (0)**	**5.2 (0.4)**
d13	*10.3 (0.9)*	**8 (0)**	**6 (2.1)**	**9.4 (1.3)**
d14	*2.4 (0.5)*	**2 (0)**	**2 (0)**	**2 (0)**
d15	*11.9 (0.3)*	**8 (0)**	**6 (0)**	**6 (0)**
d16	*5.8 (0.4)*	6 (0)	5.6 (0.8)	**3.6 (1.3)**
d17	3.4 (3.1)	2 (0)	2 (0)	2.2 (0.4)
d18	*11 (1.3)*	**3 (0)**	**2.8 (0.4)**	**2 (0)**
d19	*4.8 (0.6)*	**2 (0)**	5 (0)	**2 (0)**
d20	*7.3 (0.5)*	**4 (0)**	**4 (0)**	**4 (0)**
d21	1.1 (0.3)	1 (0)	1.2 (0.4)	1 (0)
Mean	*10.2 (1.3)*	**5.1 (0.5)**	**5.3 (0.8)**	**4.7 (0.6)**
Better	0	16	15	18
Worse	18	0	0	0

The cardinality of the final feature subset is also important. The average number of features selected for each method and its respective standard deviation is presented in table 24. Bold and italic faces have the same meaning as in table 23, as well as the lines 'Mean', 'Better' and 'Worse'. There is no condition that the GA method achieved average feature set size statistically significantly smaller than any CFS method. However, it had significantly bigger averages for 18 conditions. The smallest average number of features was achieved by the Rank-SFS-GA method, while the GA feature selection had the highest average across the conditions.

Table 25 – Training time (in seconds) of each method for each condition.

	GA	Rank-GA	SFS-GA	R-S-GA
d01	*6186 (722)*	**3186 (1020)**	**3224 (910)**	**2121 (452)**
d02	*5546 (437)*	**2173 (299)**	**2212 (216)**	**1835 (570)**
d03	*8135 (1418)*	**3006 (681)**	**2742 (468)**	**3031 (766)**
d04	*3148 (361)*	**239 (24)**	**264 (23)**	**240 (22)**
d05	*4022 (472)*	3791 (372)	**1052 (520)**	**2750 (496)**
d06	*4611 (481)*	**230 (19)**	**256 (17)**	**229 (19)**
d07	*3557 (328)*	**557 (516)**	**283 (33)**	**1080 (464)**
d08	*4137 (565)*	**1995 (219)**	**2328 (328)**	**1721 (168)**
d09	*7777 (2275)*	**3206 (474)**	**2548 (677)**	**2007 (186)**
d10	*5582 (1504)*	**844 (703)**	**683 (505)**	**714 (433)**
d11	*5055 (1447)*	**303 (27)**	**358 (44)**	**401 (35)**
d12	*4332 (595)*	**1719 (166)**	**1954 (257)**	**1779 (206)**
d13	*4582 (808)*	**1627 (148)**	**2024 (327)**	**1737 (179)**
d14	*2989 (359)*	**312 (29)**	**348 (36)**	**308 (26)**
d15	*8418 (3115)*	**2840 (246)**	**2561 (368)**	**1625 (206)**
d16	*5612 (1544)*	**3102 (337)**	**1680 (666)**	**711 (646)**
d17	*5066 (1062)*	**397 (35)**	**657 (81)**	**337 (30)**
d18	*6403 (2726)*	**528 (54)**	**518 (128)**	**692 (173)**
d19	*5195 (1910)*	**396 (34)**	**1743 (289)**	**328 (26)**
d20	*8103 (3917)*	**528 (57)**	**533 (69)**	**632 (149)**
d21	*7529 (2360)*	**244 (21)**	**318 (70)**	**246 (19)**
Mean	*5523 (1353)*	**1487 (261)**	**1347 (287)**	**1168 (251)**
Better	0	20	21	21
Worse	21	0	0	0

One theoretical advantage expected by using the CSF proposal is to reduce the time spent to select the features. Therefore, table 25 presents the average time to select the features and training the ELM classifier. Bold and italic faces have the same meaning as in table 23, as well as the lines 'Mean', 'Better' and 'Worse'. As theoretically expected, all CFS methods for all conditions are statistically significantly faster than the GA method. The only exception is the Rank-GA method for condition d05. The fastest method, in average, is the Rank-SFS-GA, which has the longest cascade of feature selection methods.

However, there are situations that the SFS-GA expent less time than the Rank-SFS-GA, e.g. condition d07.

Table 26 – F1-score of each method for each condition.

	GA	Rank-GA	SFS-GA	R-S-GA
d01	*99.77 (0.08)*	**99.83 (0.03)**	**99.86 (0.03)**	99.84 (0.05)
d02	*97.95 (0.61)*	97.66 (0.41)	**98.75 (0.47)**	98.38 (0.55)
d03	62.83 (3.22)	62.46 (3.33)	60.23 (1.89)	64.08 (2.44)
d04	99.98 (0.03)	100.0 (0.00)	99.97 (0.03)	100.0 (0.00)
d05	**99.89 (0.04)**	*78.68 (0.29)*	99.89 (0.05)	*80.50 (1.03)*
d06	97.60 (4.47)	99.91 (0.06)	99.87 (0.00)	99.87 (0.00)
d07	*99.97 (0.03)*	99.98 (0.03)	**100.0 (0.00)**	99.95 (0.03)
d08	*80.83 (6.63)*	**89.36 (0.88)**	82.18 (3.90)	76.49 (5.84)
d09	*56.93 (3.65)*	**63.58 (0.66)**	**71.81 (5.16)**	**62.50 (0.05)**
d10	*75.44 (2.43)*	**77.67 (1.39)**	**79.10 (1.71)**	75.55 (2.57)
d11	*87.25 (1.73)*	87.09 (0.00)	**88.84 (0.29)**	**88.61 (0.21)**
d12	***92.39 (1.86)***	**93.76 (0.00)**	92.70 (1.40)	*90.73 (1.15)*
d13	*77.35 (7.04)*	75.00 (0.00)	**83.66 (2.96)**	**87.11 (0.73)**
d14	99.94 (0.00)	99.94 (0.00)	99.94 (0.00)	99.94 (0.00)
d15	*63.84 (3.13)*	62.43 (0.00)	**67.53 (0.24)**	**68.08 (1.81)**
d16	*82.99 (2.96)*	**85.75 (0.00)**	82.98 (0.76)	**88.54 (4.05)**
d17	96.06 (4.46)	96.77 (0.00)	95.57 (0.60)	97.59 (0.22)
d18	***87.90 (5.31)***	**93.48 (0.00)**	**93.59 (0.51)**	*39.89 (27.52)*
d19	**92.18 (1.75)**	*88.80 (0.00)*	91.95 (0.24)	*88.47 (0.05)*
d20	88.29 (4.93)	88.81 (0.00)	88.75 (0.12)	90.39 (0.06)
d21	94.87 (9.25)	97.80 (0.00)	91.33 (13.65)	97.80 (0.00)
Mean	***87.35 (3.03)***	**87.56 (0.34)**	**88.98 (1.62)**	*85.44 (2.3)*
Better	4	7	9	5
Worse	12	2	0	4

As the test data files have an unbalanced number of samples for the normal and fault condition, another classification performance criteria was evaluated. The chosen metric is the F1-score (SOKOLOVA; LAPALME, 2009), which is more appropriate than the accuracy criterion for this kind of data set. The values obtained by each method are pointed out in table 26. Bold and italic faces have the same meaning as in table 23, as well as the lines 'Mean', 'Better' and 'Worse'. One may see that, regardless the difference of absolute values of table 23 and table 26, the results presented in these tables are very similar. For all conditions, when a CFS method is statistically significantly better than the GA in table 23, it is also significantly better then the GA in table 26. The same happened for statistically significantly worse performances. Consequently, the lines 'Better' and 'Worse' have the same values in both tables. Again, the Rank-GA and the SFS-GA configurations are significantly better than GA, and the R-S-GA method is significantly worse, in average.

The experimental results presented in this section suggest that the SFS-GA method has the best general performance, with a good trade-off considering training time, cardinality of the final feature set and classification performance. In average, it takes considerably less time than GA to select features; its final subsets have almost half of the features obtained by GA; and its average performance surpasses the other three methods both in accuracy and F1-score. However, considering that the main objective of CFS is to speed up the feature selection without loss of performance, the Rank-GA might be a interesting option too. For instance, Rank-GA has better performance than SFS-GA in all measures for condition d12. On the other hand, the R-S-GA configuration did not show good results in terms of classification performance.

5.3 Features Analysis

Table 27 compiles the features always selected in all rounds for each method. It also shows in parenthesis the total number of selected features on the ten rounds. So, one can identify if the commonly selected features are the same among the different methods, if there is a large variability of the features selected on the different rounds of each method (i.e., the number of commonly selected features are much smaller than the total number of selected features) and if there are features always selected independently of the type of fault. It may be seen that Rank-GA and R-S-GA provide no constantly selected feature for d02 on the ten rounds. For conditions d04, d11, d17 and d21, all methods present the same set of features. However, their classification performances are different. This occurs because these methods selected other features on the different rounds besides the common set of table 27. The fact that all methods have repeatedly selected the same features subsets might indicate that they are important for identifying that particular process condition. The opposite behavior happens for condition d14, where the subset is different for the methods, but the classification performance is identical. GA and SFS-GA selected feature 21 only, while Rank-GA and R-S-GA selected features 9 and 21. This behavior suggests that feature 9 is irrelevant for the classification of this condition, and the former methods were able to discard it. For the conditions were the subsets are different, usually there is an intersection among the subsets. These features should be important for classifying the observed condition. As one may expect, there is no feature subset that is common for every type of fault. However, the feature 7 is the mostly selected feature for all methods. It has been selected by GA for 9 conditions, by Rank-GA, SFS-GA and R-S-GA for 11, 7 and 7 conditions, respectively. It should be also considered important, according to table 27, the features 13, 16, 18, 19 and 20. These five features were selected by the four feature selection methods for at least four conditions.

Table 27 – Features always selected in each of the evaluation rounds for the different methods.

	GA	Rank-GA	SFS-GA	R-S-GA
d01	(42) 3 4 7 16 21 22 23 39	(29) 7 18 20	(30) 3 18	(25) 3 18 19 20
d02	(22) 3 7 13 16 20 28 37 38 39 40 41	(12)	(21) 1 3 11 30 39	(18)
d03	(11) 7 13 18 19 20 37 38 39 40 41 50	(12) 7 13 18 19 37 39 40 41 50	(12) 7 13 20 37 38 40	(12) 7 13 19 37 40
d04	(37) 51	(1) 51	(1) 51	(1) 51
d05	(45) 17 18 22 52	(10) 7 16 18 20 38 50 52	(6) 11 17 52	(9) 18 20 52
d06	(48) 1 44	(1) 44	(1) 44	(1) 44
d07	(42) 3 45	(4) 45	(3) 45	(4) 7 45
d08	(22) 1 4 7 16 20	(5) 1 20 23 38 44	(12) 1 16 20 44 46	(9) 1 20 23 29 44
d09	(11) 7 13 16 18 19 20 37 38 39 40 50	(6) 7 13 18 19 20 50	(6) 7 13 18 20 50	(7) 7 13 18 20 50
d10	(7) 16 18 19 38 50	(6) 7 18 19 38 50	(7) 16 18 19 38 50	(7) 7 18 19 38 50
d11	(4) 9 51	(2) 9 51	(2) 9 51	(2) 9 51
d12	(18) 4 13 20	(7) 7 11 13 19 20 31 50	(10) 4 7 11 13 16 20 31 46	(7) 11 16 20 50
d13	(20) 7 16 19 21 38	(8) 7 11 13 16 18 21 34 50	(11) 13 18 19 50	(11) 7 13 16 18 19 50
d14	(3) 21	(2) 9 21	(3) 21	(2) 9 21
d15	(12) 7 13 16 18 19 20 37 38 39 41 50	(8) 7 13 16 18 19 39 40 50	(6) 7 16 18 19 20 50	(6) 7 13 16 18 39 50
d16	(6) 13 19 20 50	(6) 7 13 18 19 20 50	(6) 7 18 19 50	(6) 18 19 50
d17	(13) 9 21	(2) 9 21	(2) 9 21	(3) 9 21
d18	(22) 4 7 22 37 39 40 41	(3) 4 7 11	(3) 4 22	(4)
d19	(6) 5 46	(2) 5 46	(5) 5 7 13 16 46	(2) 5 46
d20	(10) 7 11 13 20 46	(4) 7 13 20 46	(4) 7 13 20 46	(4) 7 13 20 46
d21	(2) 45	(1) 45	(2) 45	(1) 45

Chapter Considerations

This chapter presented the concept of Cascade Feature Selection (CFS) where simple or complex feature selection methods are applied sequentially. Experiments were performed with the data produced by the Tennessee Eastman Chemical Process Simulator and the Extreme Learning Machine (ELM) as classifier model. Hybrid versions of ranking and SFS algorithms, that mix filter and wrapper approaches, were combined with a pure wrapper feature selection based on the Genetic Algorithm. Three CFS configurations were tested, i.e. Rank-GA, SFS-GA and R-S-GA. Results suggest that the cascade arrangement can produce smaller final feature subsets, expending less time, with higher classification performances than the feature selection based on the Genetic Algorithm. The three CFS methods significantly reduce the final number of the selected features and the computational effort. For the TE data, two CFS combinations, Rank-GS and SFS-GA presented a statistically significant improvement in performance classification. The SFS-GA was considered the best trade-off, considering the final number of features, speed and classification performance.

6 One vs. One Ensemble Feature Selection

The combination of heterogeneous feature extraction models and feature selection methods has shown promissory results in automatic fault diagnosis (RAUBER; BOLDT; VAREJÃO, 2015; BROETTO; VAREJÃO, 2016). However, the optimal set of features to identify multiple conditions might be different, depending of which conditions are considered. Here a classification scheme is presented where the features are selected by pairs of conditions, aiming to maximize the identification of each class with distinct feature sets. In order to implement this idea, a one-versus-one performance estimation arrangement has to be defined. Nevertheless, even ranking feature selection might be expensive when large datasets (as the ESP dataset) are studied. Therefore, techniques like Error-Correcting Output Codes (ECOC) (DIETTERICH; BAKIRI, 1995) used with feature selection for large datasets may be excessively expensive. For instance, table 28 shows a one-versus-one code design for five classes, where $\mathcal{C}$ represents classes and $\mathcal{L}$ represents learners, or classifiers. Let M be the coding design matrix of table 28, with elements m_{kl}, and s_l be the predicted classification score, or confidence, for the positive class of learner l. Five classes ($K = 5$) demand $K \times (K-1)/2 = 10$ learners in a one-versus-one arrangement. ECOC solves an optimization problem for each sample to give the final prediction, as shown in equation 6.1, where $g(\cdot,\cdot)$ measures confidence of l learner for label k, that can be positive or negative. The value of $\hat{k}$ returned from equation 6.1 is the value of k that minimizes the expression $\frac{\sum_{l=1}^{L}|m_{kl}|g(m_{kl},s_l)}{\sum_{l=1}^{L}|m_{kl}|}$, that calculates the average discordance of the learners for label k. Therefore, the label with the minimal mean discordance among the learners is chosen as the predicted label.

$$\hat{k} = \underset{k}{\operatorname{argmin}} \frac{\sum_{l=1}^{L}|m_{kl}|g(m_{kl},s_l)}{\sum_{l=1}^{L}|m_{kl}|} \tag{6.1}$$

For a dataset with 4570 samples, it takes a long time to perform multiple cross-validations which have comparable results considering statistical significance, since each sample would have to perform an optimization over matrix M. Whence, a simpler one-versus-one scheme is proposed here.

Table 28 – One-versus-one code design for five classes.

	$\mathcal{L}_1$	$\mathcal{L}_2$	$\mathcal{L}_3$	$\mathcal{L}_4$	$\mathcal{L}_5$	$\mathcal{L}_6$	$\mathcal{L}_7$	$\mathcal{L}_8$	$\mathcal{L}_9$	$\mathcal{L}_{10}$
$\mathcal{C}_1$	1	1	1	1	0	0	0	0	0	0
$\mathcal{C}_2$	-1	0	0	0	1	1	1	0	0	0
$\mathcal{C}_3$	0	-1	0	0	-1	0	0	1	1	0
$\mathcal{C}_4$	0	0	-1	0	0	-1	0	-1	0	1
$\mathcal{C}_5$	0	0	0	-1	0	0	-1	0	-1	-1

In the proposed method the learners $\mathcal{L}$ answer the actual label $\mathcal{C}$ instead of a code in M. Thus, the learners design is arranged as shown in table 29. Let P be the prediction matrix with elements p_l, and s_l be the predicted classification confidence of the learner prediction. The label with the maximal average confidence is returned as the predicted label by the ensemble, as shown in equation 6.2, where L is the quantity of labels.

$$\hat{C} = \max_{\mathcal{C}} \frac{\sum_{l=1}^{L} \begin{cases} s_l, & p_l = \mathcal{C} \\ 0, & \text{otherwise} \end{cases}}{\sum_{l=1}^{L} \begin{cases} 1, & p_l = \mathcal{C} \\ 0, & \text{otherwise} \end{cases}} \tag{6.2}$$

No optimization needs to be performed and the calculus of the predicted label can be done by efficient matrix calculations present in many software libraries, like e.g. Matlab. Preliminary experiments have shown better classification results for average confidence than for the sum of confidence, because the average confidence rewards the highest confidence levels instead of the highest number of votes. It is expected that learners do not give high scores for samples with actual labels that were not present in the classifiers training dataset. Hence, a low score reduces the average confidence for that label.

Table 29 – Proposed one-versus-one approach for five classes.

$\mathcal{L}_1$	$\mathcal{L}_2$	$\mathcal{L}_3$	$\mathcal{L}_4$	$\mathcal{L}_5$	$\mathcal{L}_6$	$\mathcal{L}_7$	$\mathcal{L}_8$	$\mathcal{L}_9$	$\mathcal{L}_{10}$
$\mathcal{C}_1$	$\mathcal{C}_1$	$\mathcal{C}_1$	$\mathcal{C}_1$	$\mathcal{C}_2$	$\mathcal{C}_2$	$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_3$	$\mathcal{C}_4$
$\mathcal{C}_2$	$\mathcal{C}_3$	$\mathcal{C}_4$	$\mathcal{C}_5$	$\mathcal{C}_3$	$\mathcal{C}_4$	$\mathcal{C}_5$	$\mathcal{C}_4$	$\mathcal{C}_5$	$\mathcal{C}_5$

The training and test procedure of the proposed approach is presented in figure 37. Right angle rectangles represent datasets, round corner rectangles represent processes and ellipses represent classifiers or predictions. The training process is represented by the largest dashed rectangle. This process splits the training dataset into smaller datasets with samples of only two classes each. Considering that the ESP dataset has $n = 5$ conditions, $n * (n-1)/2 = 10$ datasets have to be assembled. Feature selection is performed over each binary dataset generating potentially reduced datasets, probably with different features. Classifiers that are trained with these datasets compose the ensemble classifier. The test process, represented by the smaller dashed rectangle, receives the test dataset and the ensemble classifier. The prediction label process performed by the ensemble classifier is represented by the dotted rectangle inside the test process. Each classifier in the ensemble uses the whole test dataset to predict the label of each sample. Obviously, many of those predictions will be wrong, since the classifiers in the ensemble were not trained with all classes. However, it is expected that classifiers that try to identify samples of classes with which they were not trained, give a low score for those samples. Then, the low score reduces the average confidence for the wrong predicted label. The label with the highest average confidence is chosen as the predicted label.

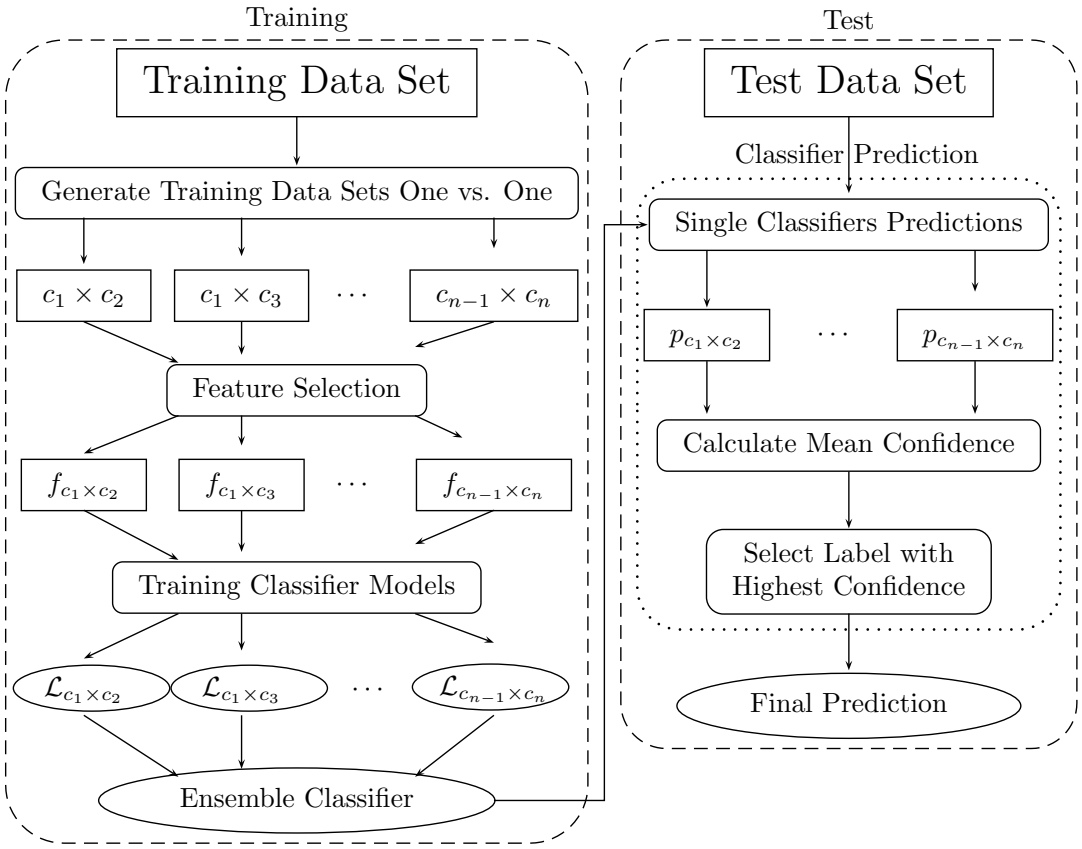


Figure 37 – One vs. One Feature Selection.

6.1 F-Measure Performance

Figure 38 shows the boxplot chart of the five methods compared, where KNN is the 1-NN classifier using the original ESP dataset from OLIVEIRA-SANTOS et al. (2016), KNN+ is the 1-NN classifier using the extended dataset, that has the same features of OLIVEIRA-SANTOS et al. (2016) plus the twelve new features. KNN+R is the 1-NN classifier using the extended dataset and a conventional multiclass ranking feature selection. KNN+ER is the one vs. one ensemble feature selection approach using the extended dataset with the 1-NN classifier with a ranking feature selection. KNN+ECR is the one vs. one ensemble feature selection approach using the extended dataset, the 1-NN classifier and a cascade ranking feature selection configuration. Since ranking is an efficient feature selection method, the cascade configuration here uses another ranking algorithm, applied as filter, to select 18 features. The three methods that use feature selection used the ranking approach applied as wrapper. The internal validation was the average of five rounds of a holdout validation with division 70% for training and 30% for test. In the case of KNN+ECR, 18 features were selected in the first stage of the cascade. Its second stage used the ranking feature selection with the same configuration of KNN+R and KNN+ER. By visual inspection it is easy to see the improvement provided by the conjunction of additional features and the proposed ensemble approach.

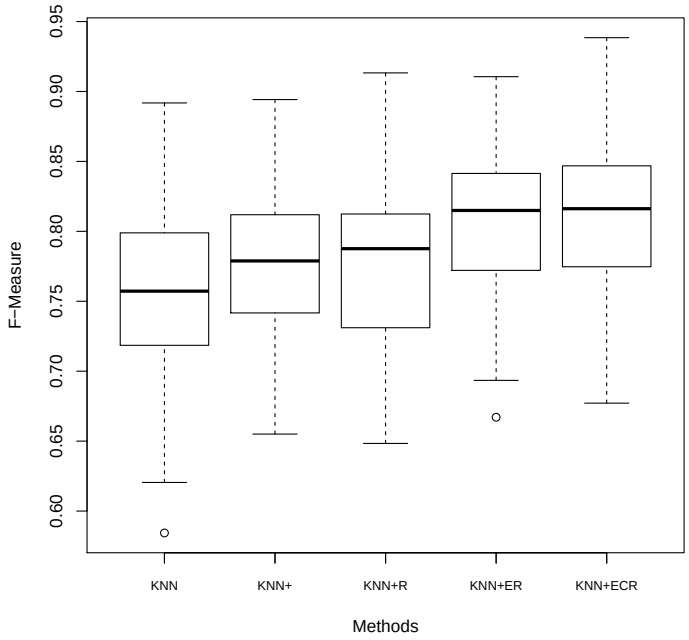


Figure 38 – F-Measure boxplot.

Table 30 numerically repeats the parameters used to plot the chart of figure 38, where 'Min.' is the minimum value, 'Q_1' is the first quartile, 'Q_3' is the third quartile and 'Max.' is the maximum value obtained for each classification method. The highest values are in bold face.

Table 30 – Parameters of the boxplot in figure 38.

	KNN	KNN+	KNN+R	KNN+ER	KNN+ECR
Min.	0.5843	0.6550	0.6483	0.6671	**0.6771**
Q_1	0.7198	0.7422	0.7312	0.7722	**0.7753**
Median	0.7572	0.7788	0.7876	0.8149	**0.8162**
Mean	0.7563	0.7758	0.7752	0.8049	**0.8109**
Q_3	0.7977	0.8115	0.8124	0.8406	**0.8468**
Max.	0.8918	0.8942	0.9133	0.9106	**0.9385**

6.2 Statistical Significance

Figure 39 shows a matrix to compare the five methods by pairs. In the lower triangle, the histograms of the differences of the F-measures for each pair of method are shown. The zero of the horizontal axis is in the center of each histogram box. The upper triangle presents the corresponding p-values of the correlated t-test for each pair of methods. The value in bold face, 0.005331 in the top right square, indicates a statistically significant difference. It means that the KNN+ECS method is statistically significant better than the KNN method.

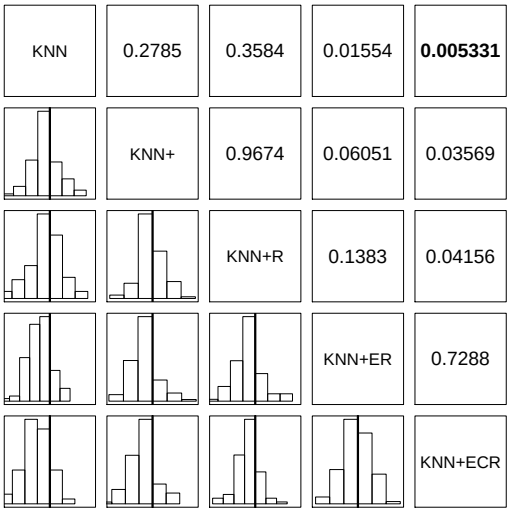


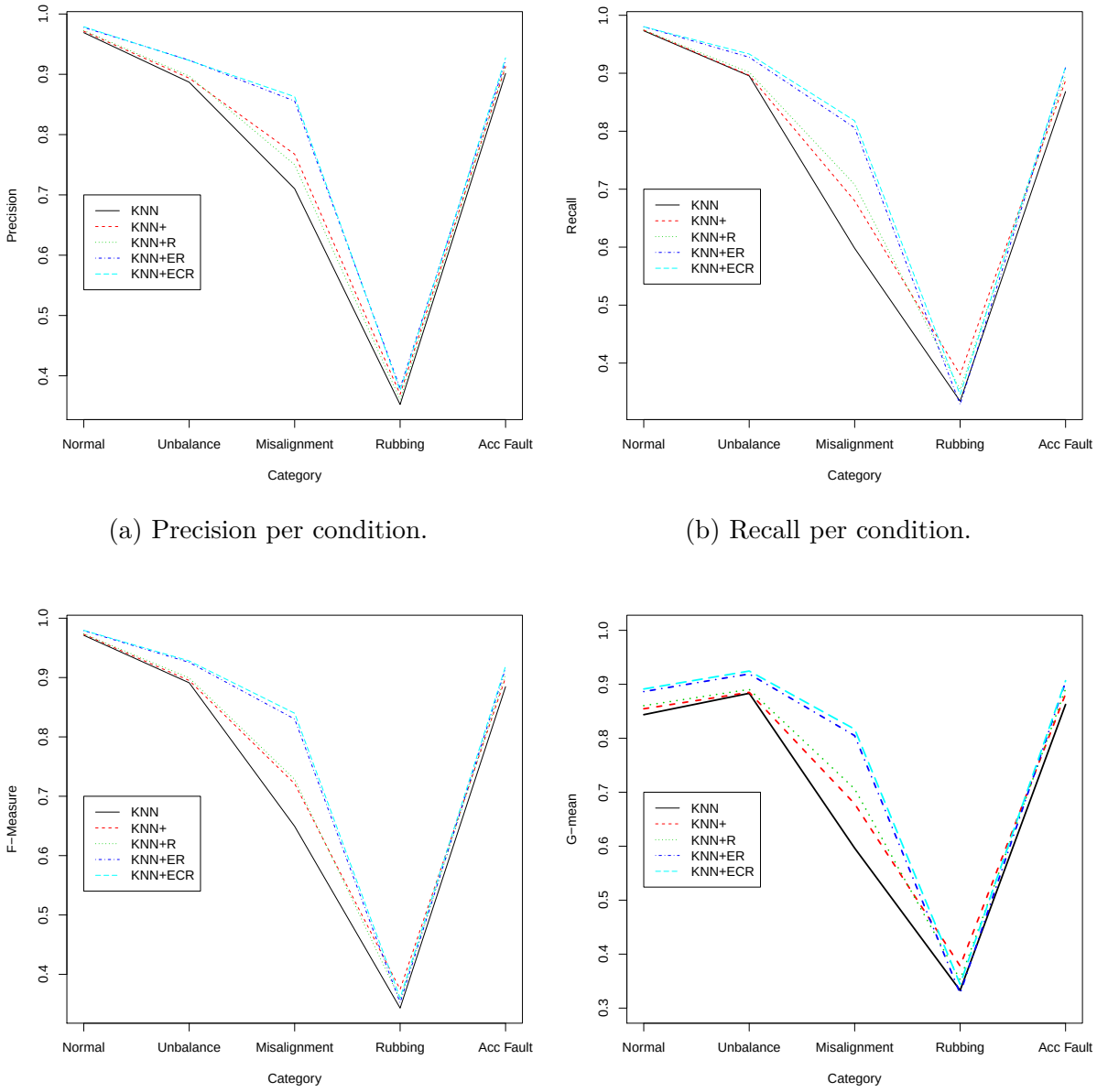
Figure 39 – Pairwise comparison between methods. The lower triangle shows the histograms of the differences of the Fscores for each pair of method. The upper triangle presents the corresponding p-values of the correlated t-test.

It is important to remark each method improves the method before. However, only two methods have statistical significant difference, that are the 1-NN with the original dataset and the method that uses the three proposals presented of this work together.

6.3 Performance Classification for Each Condition

It is also important to know how the behavior of each method relates to each condition. Figure 40b shows such an information. It can be seen that the additional features improve the recall for all conditions, when compared with the original dataset. When the methods KNN+R, KNN+ER and KNN+ECR are compared with KNN+, they improve the recall for all conditions, except for the rubbing fault. This behavior can theoretically be expected. According to KUNCHEVA; WHITAKER (2003), a classifier ensemble only can improve the classification if the individual performance of each learner is higher than 0.5, otherwise, it tends to decrease performance. This happens for the rubbing condition, where multiclass classifiers have poor classification performances for this class.

Knowing that rubbing is the most difficult condition to be identified, two research approaches can be envisioned. Firstly, rubbing is the condition with less examples in the dataset. Only 35 of the 4570 samples are labeled as rubbing. Oversampling techniques might solve this problem. Secondly, the used features are not discriminative enough to discern rubbing properly, and more features have to be extracted and tested. The new features might be designed exclusively for this condition or a range of generic features can be extracted and filter through feature selection methods.



(a) Precision per condition.

(b) Recall per condition.

(c) F-Measure per condition.

(d) G-Mean per condition.

Figure 40 – Performance classification for each condition.

6.4 Comparison with Well Known Ensembles

The method presented here is also compared with two well known ensembles methods, i.e. Random Forest and Adaboost. However, because of the randomness of the inner validation for feature selection, there is no guarantee that the subset of feature selected is the best subset for the test folds. Therefore, each binary classifier is compounded of five classifiers with potentially different feature sets, which generates the desirable diversity among the classifiers of an ensemble. Figure 41 shows the boxplot comparing these three ensembles methods. It can be seen a slight higher performance of the KNN ensemble.

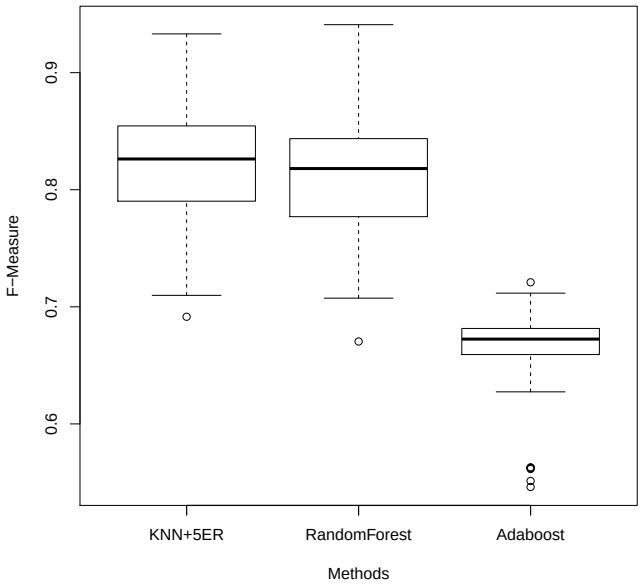


Figure 41 – F-Measure boxplot for ESP dataset.

Figure 42 shows the statistical comparison of the classification methods. Both, the KNN ensemble and the Random Forest were statistically significant better than the Adaboost algorithm. On the other hand, there is no statistical significant difference between the KNN ensemble and Random Forest.

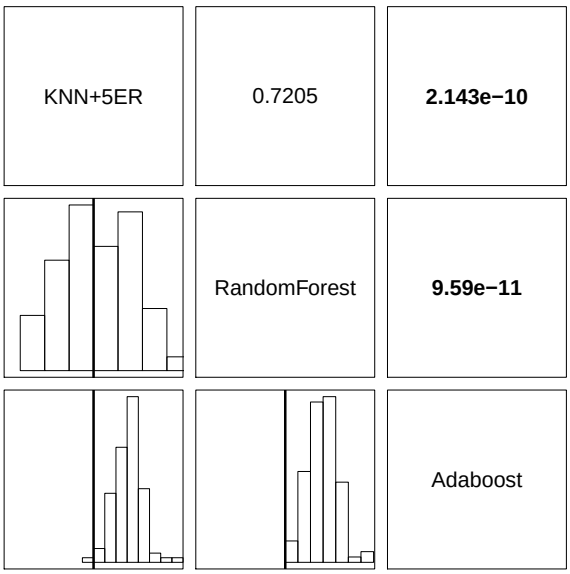
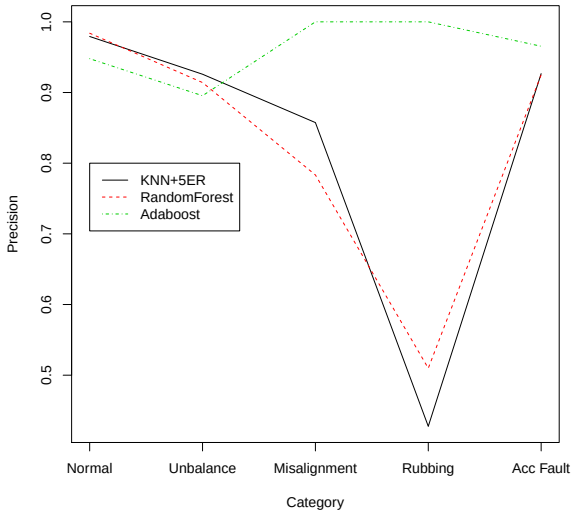
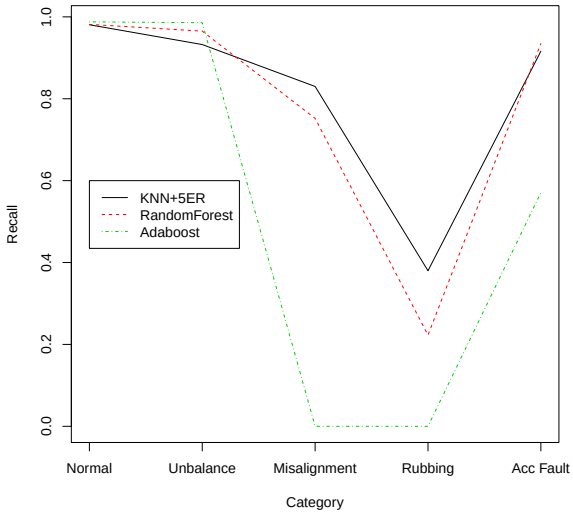


Figure 42 – Pairwise comparison between methods. The lower triangle shows the histograms of the differences of the F-scores for each pair of method. The upper triangle presents the corresponding p-values of the correlated t-test.

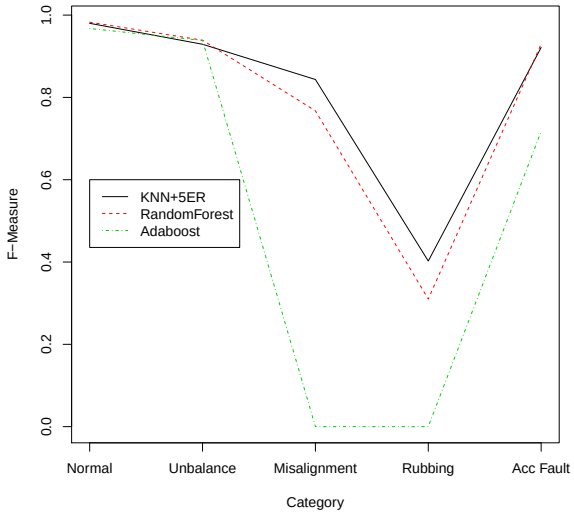
Figure 43 shows performances of each ensemble method for each class individually. It can be seen that the recall for the Adaboost algorithm is zero for misalignment and rubbing. Consequently, the F-measure and the G-mean are also zero for these conditions. This explains the low performance of Adaboost compared with the other ensembles. In general, the KNN ensemble has a better classification performance than Random Forest for these two difficult classes.



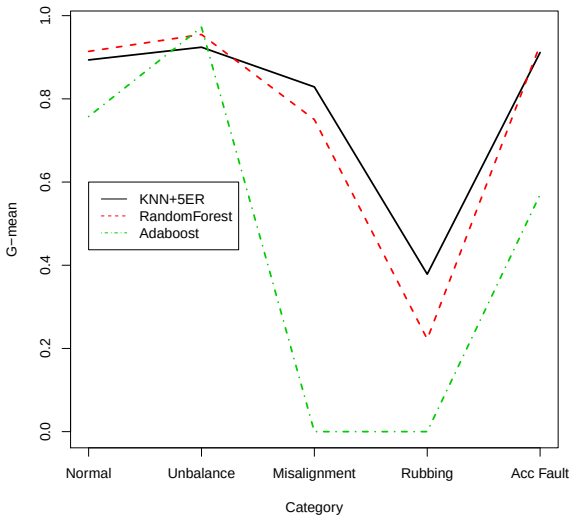
(a) Precision per condition.



(b) Recall per condition.



(c) F-Measure per condition.



(d) G-Mean per condition.

Figure 43 – Performance classification for each condition.

6.5 Statistical Significance over CWRU and ESP datasets

Beyond the statistical significance of the methods taking in account a single dataset, it is also presented in table 31 a comparison of a single 1-NN classifier and the three ensemble methods of the previous section over the three main multi-class datasets of this work, i.e. ESP dataset, CWRU dataset and Special AUC-ROC CWRU dataset. The statistical method proposed in CORANI et al. (2016) was used to estimate the statistical significance of the difference between each pair of the tested methods. This statistical method demands repeated paired cross-validation, since the metric value estimated for each fold is compared. Ten rounds were performed.

Table 31 – Probability of one method be better than other.

Classifiers	Left	/	Rope	/	Right
KNN vs. KNN+5ER	0.1849	/	0.0037	/	0.8114
KNN vs. Random Forest	0.2349	/	0.0115	/	0.7536
KNN vs. Adaboost	0.8575	/	0	/	0.1425
KNN+5ER vs. Random Forest	0.4116	/	0.3847	/	0.2037
KNN+5ER vs. Adaboost	0.8781	/	0	/	0.1219
Random Forest vs. Adaboost	0.8744	/	0	/	0.1256

It can be seen that the KNN ensemble has higher probability to be better than Adaboost and Random Forest. Due the low performance of Adaboost, even the 1-NN algorithm alone has higher probability to be better. The tendency of Adaboost to overfitting (QUINLAN, 1996) brings down its performance with ESP dataset for conditions with relative small amount of samples, i.e. misalignment and rubbing.

Chapter Considerations

The union of a one-versus-one classification approach bundled with feature selection of an extended dataset has shown statistically significant improvement in classification performance. The additional features improved the overall classification, but without statistical significance. Preliminary feature selection experiments were performed with the original and extended datasets. All experiments using feature selection with multiclass classifiers reduced the classification performance. Only the one-versus-one arrangement was able to improve the classification scores significantly. This result suggests that different feature sets have significantly different performances to identify each group of process conditions, corroborating the fundamental hypothesis of this work. As the proposed approach did not improve the classification of the rubbing fault, future works will try oversampling and extract new features to improve the rubbing condition classification performance.

Conclusion

This work have presented a classifier ensemble feature selection method based on three principles, i.e. Heterogeneous Feature Models (RAUBER; BOLDT; VAREJÃO, 2015), Cascade Feature Selection (BOLDT; RAUBER; VAREJÃO, 2017) and One vs. One Ensemble Feature Selection (BOLDT et al., 2017). Experiments that corroborate the effectiveness of each principle have been presented in chapters 4, 5 and 6, respectively. Chapter 4 showed that using Heterogeneous Feature Models is not always enough to promote a better discriminative power to diagnosis systems. Only after a posterior information filter using a feature selection method, the general performance can be expected to improve. Chapter 5 presented the Cascade Feature Selection concept, where simple or complex feature selection methods are applied sequentially. Three CFS configurations were tested, i.e. Rank-GA, SFS-GA and R-S-GA. Results suggest that the cascade arrangement can produce smaller final feature subsets, expending less time, with higher classification performances than a feature selection based on Genetic Algorithm. Chapter 6 merged the idea of one-versus-one classification approach with Cascade Feature Selection, from chapter 5 and Heterogeneous Feature Models, from chapter 4. Experiments have shown statistically significant improvement in classification performance. Each one of the presented methods have their value by their own. However, as shown in chapter 6, the union of these techniques can improve the classification performance of a diagnosis system significantly.

The development of this work have faced many challenges. Despite the variety of papers dealing with the fault diagnostic problem, most of the public available datasets have some issues, including those presented here. These issues are reflected as limitations of this work. The Case Western Reserve University (CWRU) dataset presented in section 3.1 has vibratory data acquired from bearings with artificially induced failures. Consequently, the failures identified in each vibratory data shows a constant behavior not present in bearings which the failures appears naturally. Therefore, there is no guarantee that the automatic fault diagnostic system trained with this dataset will perform well in the production environment. The Tennessee Eastman Chemical Process (TE) dataset presented in section 3.2 had its data produced by a simulator. Then, the diagnostic system performance trained with this data may have a bigger difference from the reality than the system trained with the CWRU dataset. The Electric Submersible Pump (ESP) dataset in section 3.3 does not have the issues of the previous datasets. However, the tests which the vibratory data is acquired is made a a water tank while the use of the ESPs for production is in oil wells. The water viscosity is completely different of oil. Thus, even that labels given by the specialist may not fit with the reality. In future works, a generation of real datasets without the presented limitations is envisioned.

As Deep Neural Networks, also known as Deep Learning (MNIH et al., 2015), has called the attention of the scientific community, Convolutional Neural Networks has also been used for fault diagnosis (HOANG; KANG, 2017). Using these state of art it is possible to make a diagnostic system represented by the continuous line in figure 1. Therefore, such a continuous learning system, able to extract, select, training classifiers and analyze the features combination, is now possible using the mentioned techniques. The Convolutional Neural Networks can extract features without predetermined models. Consequently, low discriminant features can be excluded an new feature might be extracted. This will be the next challenge that we will endeavor to overcame.

References

BOLDT, F. de A.; RAUBER, T. W.; OLIVEIRA-SANTOS, T.; RODRIGUES, A.; VAREJÃO, F. M.; RIBEIRO, M. P. Binary feature selection classifier ensemble for fault diagnosis of submersible motor pump. In: *2017 IEEE 26th International Symposium on Industrial Electronics (ISIE)*. [S.l.: s.n.], 2017. Cited 8 times in pages 15, 16, 18, 19, 34, 39, 65, and 101.

BOLDT, F. de A.; RAUBER, T. W.; VAREJÃO, F. M. Feature extraction and selection for automatic fault diagnosis of rotating machinery. In: *X Encontro Nacional de Inteligencia Artificial e Computacional (ENIAC).-Fortaleza, Ceará*. [S.l.: s.n.], 2013. p. 213–220. Cited in page 20.

BOLDT, F. de A.; RAUBER, T. W.; VAREJÃO, F. M. Evaluation of the extreme learning machine for automatic fault diagnosis of the tennessee eastman chemical process. In: IEEE. *Industrial Electronics Society, IECON 2014-40th Annual Conference of the IEEE*. [S.l.], 2014. p. 2551–2557. Cited 3 times in pages 20, 52, and 58.

BOLDT, F. de A.; RAUBER, T. W.; VAREJÃO, F. M. A fast feature selection algorithm applied to automatic faults diagnosis of rotating machinery. *Journal of Applied Computing Research*, v. 3, n. 2, p. 78–86, 2014. Cited 2 times in pages 15 and 20.

BOLDT, F. de A.; RAUBER, T. W.; VAREJÃO, F. M. Single sequence fast feature selection for high-dimensional data. In: IEEE. *Tools with Artificial Intelligence (ICTAI), 2015 IEEE 27th International Conference on*. [S.l.], 2015. p. 697–704. Cited 3 times in pages 19, 38, and 39.

BOLDT, F. de A.; RAUBER, T. W.; VAREJÃO, F. M. Cascade feature selection and elm for automatic fault diagnosis of the tennessee eastman process. *Neurocomputing*, v. 239, p. 238 – 248, 2017. ISSN 0925-2312. Cited 8 times in pages 15, 17, 19, 39, 41, 52, 77, and 101.

BOLDT, F. de A.; RAUBER, T. W.; VAREJÃO, F. M.; RIBEIRO, M. P. Performance analysis of extreme learning machine for automatic diagnosis of electrical submersible pump conditions. In: IEEE. *Industrial Informatics (INDIN), 2014 12th IEEE International Conference on*. [S.l.], 2014. p. 67–72. Cited 2 times in pages 19 and 57.

BOLDT, F. de A.; RAUBER, T. W.; VAREJÃO, F. M.; RIBEIRO, M. P. Fast feature selection using hybrid ranking and wrapper approach for automatic fault diagnosis of motorpumps based on vibration signals. In: IEEE. *Industrial Informatics (INDIN), 2015 IEEE 13th International Conference on*. [S.l.], 2015. p. 127–132. Cited 5 times in pages 15, 16, 19, 41, and 78.

BOUCKAERT, R. R. Estimating replicability of classifier learning experiments. In: ACM. *Proceedings of the twenty-first international conference on Machine learning*. [S.l.], 2004. p. 15. Cited 2 times in pages 16 and 32.

BRAGA, A. de P.; CARVALHO, A. C. P. d. L. F. d.; LUDERMIR, T. B. *Redes neurais artificiais: teoria e aplicações*. [S.l.]: LTC Editora, 2007. Cited in page 27.

BREIMAN, L. Bagging predictors. *Machine learning*, Springer, v. 24, n. 2, p. 123–140, 1996. Cited in page 42.

BREIMAN, L. Random forests. *Machine learning*, Springer, v. 45, n. 1, p. 5–32, 2001. Cited in page 43.

BROETTO, R. S.; VAREJÃO, F. M. Heterogeneous feature models and feature selection applied to detection of street lighting lamps types and wattages. In: IEEE. *Industrial Electronics Society, IECON 2016-42nd Annual Conference of the IEEE*. [S.l.], 2016. p. 933–938. Cited 3 times in pages 17, 65, and 91.

BURGES, C. J. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, Springer, v. 2, n. 2, p. 121–167, 1998. Cited 2 times in pages 28 and 43.

CAO, J.; CHEN, T.; FAN, J. Landmark recognition with compact bow histogram and ensemble elm. *Multimedia Tools and Applications*, Springer, v. 75, n. 5, p. 2839–2857, 2016. Cited in page 44.

CHEBIL, J.; NOEL, G.; MESBAH, M.; DERICHE, M. Wavelet decomposition for the detection and diagnosis of faults in rolling element bearings. *Jordan Journal of Mechanical and Industrial Engineering*, Hashemite University, v. 3, n. 4, p. 260–267, 2009. Cited in page 49.

CHIANG, L.; BRAATZ, R.; RUSSELL, E. *Fault Detection and Diagnosis in Industrial Systems*. Springer London, 2001. (Advanced Textbooks in Control and Signal Processing). ISBN 9781852333270. Available from Internet: <http://web.mit.edu/braatzgroup/links-.html>. Cited 2 times in pages 52 and 53.

COIFMAN, R. R.; WICKERHAUSER, M. V. Entropy-based algorithms for best basis selection. *IEEE Transactions on information theory*, IEEE, v. 38, n. 2, p. 713–718, 1992. Cited in page 49.

CORANI, G.; BENAVOLI, A.; DEMŠAR, J.; MANGILI, F.; ZAFFALON, M. Statistical comparison of classifiers through bayesian hierarchical modelling. *arXiv preprint arXiv:1609.08905*, 2016. Cited 3 times in pages 17, 37, and 99.

COVER, T.; HART, P. Nearest neighbor pattern classification. *Information Theory, IEEE Transactions on*, IEEE, v. 13, n. 1, p. 21–27, 1967. Cited 2 times in pages 24 and 44.

CWRU. *Case Western Reserve University, Bearing Data Center*. 2017. Http://csegroups.case.edu/bearingdatacenter/pages/welcome-case-western-reserve-university-bearing-data-center-website. Accessed: 2017-05-29. Cited in page 45.

DAVIS, J.; GOADRICH, M. The relationship between precision-recall and roc curves. In: ACM. *Proceedings of the 23rd international conference on Machine learning*. [S.l.], 2006. p. 233–240. Cited in page 36.

DEMŠAR, J. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine learning research*, v. 7, n. Jan, p. 1–30, 2006. Cited in page 34.

DIAO, R.; CHAO, F.; PENG, T.; SNOOKE, N.; SHEN, Q. Feature selection inspired classifier ensemble reduction. *Cybernetics, IEEE Transactions on*, IEEE, v. 44, n. 8, p. 1259–1268, 2014. Cited in page 43.

DIETTERICH, T. G.; BAKIRI, G. Solving multiclass learning problems via error-correcting output codes. *Journal of artificial intelligence research*, v. 2, p. 263–286, 1995. Cited 2 times in pages 18 and 91.

DOWNS, J. J.; VOGEL, E. F. A plant-wide industrial process control problem. *Computers & Chemical Engineering*, Elsevier, v. 17, n. 3, p. 245–255, 1993. Cited 5 times in pages 52, 53, 54, 55, and 56.

DUDA, R. O.; HART, P. E.; STORK, D. G. *Pattern Classification*. 2nd. ed. New York: John Wiley and Sons, 2001. Cited 4 times in pages 23, 25, 27, and 29.

EBERHART, R. C.; SHI, Y. *Computational intelligence: concepts to implementations*. [S.l.]: Elsevier, 2011. Cited 2 times in pages 26 and 27.

FREUND, Y.; SCHAPIRE, R. Experiments with a new boosting algorithm. In: *Proc. 13th International Conference on Machine Learning (ICML'96)*. Bari, Italy: [s.n.], 1996. Cited in page 42.

GAO, R. X.; YAN, R. *Wavelets: Theory and applications for manufacturing*. [S.l.]: Springer Science & Business Media, 2010. Cited in page 49.

GAO, X.; HOU, J. An improved svm integrated gs-pca fault diagnosis approach of tennessee eastman process. *Neurocomputing*, Elsevier, v. 174, p. 906–911, 2016. Cited 5 times in pages 13, 52, 81, 82, and 83.

GHIMIRE, D.; LEE, J. Extreme learning machine ensemble using bagging for facial expression recognition. *JIPS*, v. 10, n. 3, p. 443–458, 2014. Cited in page 44.

GUYON, I.; ELISSEEFF, A. An introduction to variable and feature selection. *The Journal of Machine Learning Research*, JMLR. org, v. 3, p. 1157–1182, 2003. Cited 2 times in pages 38 and 43.

HAN, M.; LIU, B. Ensemble of extreme learning machine for remote sensing image classification. *Neurocomputing*, Elsevier, v. 149, p. 65–70, 2015. Cited in page 44.

HE, Y.-L.; GENG, Z.-Q.; ZHU, Q.-X. Data driven soft sensor development for complex chemical processes using extreme learning machine. *Chemical Engineering Research and Design*, Elsevier, v. 102, p. 1–11, 2015. Cited in page 52.

HOANG, D.-T.; KANG, H.-J. Convolutional neural network based bearing fault diagnosis. In: SPRINGER. *International Conference on Intelligent Computing*. [S.l.], 2017. p. 105–111. Cited in page 102.

HOLM, S. A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics*, JSTOR, p. 65–70, 1979. Cited in page 37.

HUANG, G.-B.; ZHU, Q.-Y.; SIEW, C.-K. Extreme learning machine: theory and applications. *Neurocomputing*, Elsevier, v. 70, n. 1, p. 489–501, 2006. Cited in page 30.

KUBAT, M.; MATWIN, S. et al. Addressing the curse of imbalanced training sets: one-sided selection. In: NASHVILLE, USA. *ICML*. [S.l.], 1997. v. 97, p. 179–186. Cited in page 35.

KUDO, M.; SKLANSKY, J. Comparison of algorithms that select features for pattern classifiers. *Pattern recognition*, Elsevier, v. 33, n. 1, p. 25–41, 2000. Cited in page 38.

KUNCHEVA, L. I.; WHITAKER, C. J. Measures of diversity in classifier ensembles and their relationship with the ensemble accuracy. *Machine learning*, Springer, v. 51, n. 2, p. 181–207, 2003. Cited in page 95.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep learning. *Nature*, Nature Research, v. 521, n. 7553, p. 436–444, 2015. Cited in page 28.

LIU, B. Selection of wavelet packet basis for rotating machinery fault diagnosis. *Journal of Sound and Vibration*, Elsevier, v. 284, n. 3, p. 567–582, 2005. Cited in page 49.

LIU, J. Shannon wavelet spectrum analysis on truncated vibration signals for machine incipient fault detection. *Measurement Science and Technology*, IOP Publishing, v. 23, n. 5, p. 055604, 2012. Cited 3 times in pages 45, 46, and 49.

LUO, J.; YU, D.; LIANG, M. A kurtosis-guided adaptive demodulation technique for bearing fault detection based on tunable-q wavelet transform. *Measurement Science and Technology*, IOP Publishing, v. 24, n. 5, p. 055009, 2013. Cited in page 49.

MCCULLOCH, W. S.; PITTS, W. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, Springer, v. 5, n. 4, p. 115–133, 1943. Cited in page 25.

MCINERNY, S. A.; DAI, Y. Basic vibration signal processing for bearing fault detection. *IEEE Transactions on Education*, v. 46, n. 1, p. 149–156, 2003. Cited in page 50.

MINSKY, M.; PAPERT, S. Perceptrons. MIT press, 1969. Cited in page 27.

MIT, M. I. of T. *Fault Detection and Diagnosis in Industrial Systems – datasets*. 2001. Http://web.mit.edu/braatzgroup/links.html. Accessed: 2016-06-08. Cited 2 times in pages 52 and 83.

MNIH, V.; KAVUKCUOGLU, K.; SILVER, D.; RUSU, A. A.; VENESS, J.; BELLEMARE, M. G.; GRAVES, A.; RIEDMILLER, M.; FIDJELAND, A. K.; OSTROVSKI, G. et al. Human-level control through deep reinforcement learning. *Nature*, Nature Research, v. 518, n. 7540, p. 529–533, 2015. Cited in page 102.

MOBLEY, R. K. *Root Cause Failure Analysis (Plant Engineering Maintenance Series)*. [S.l.]: Butterworth-Heinemann, 1999. Cited 2 times in pages 49 and 51.

NADEAU, C.; BENGIO, Y. Inference for the generalization error. *Machine Learning*, Springer, v. 52, n. 3, p. 239–281, 2003. Cited 2 times in pages 36 and 37.

OLIVEIRA-SANTOS, T.; RAUBER, T. W.; VAREJÃO, F. M.; MARTINUZZO, L.; OLIVEIRA, W. Submersible motor pump fault diagnosis system: A comparative study of classification methods. In: IEEE. *Tools with Artificial Intelligence (ICTAI), 2016 IEEE 28th International Conference on*. [S.l.], 2016. Cited 9 times in pages 17, 25, 34, 36, 37, 58, 59, 61, and 93.

OPITZ, D. W. Feature selection for ensembles. In: *AAAI/IAAI*. [S.l.: s.n.], 1999. p. 379–384. Cited in page 43.

PARK, C. H.; KIM, S. B. Sequential random k-nearest neighbor feature selection for high-dimensional data. *Expert Systems with Applications*, Elsevier, v. 42, n. 5, p. 2336–2342, 2015. Cited 2 times in pages 32 and 44.

PENG, H.; LONG, F.; DING, C. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, IEEE, v. 27, n. 8, p. 1226–1238, 2005. Cited 4 times in pages 38, 40, 78, and 79.

PUDIL, P.; NOVOVIČOVÁ, J.; KITTLER, J. Floating search methods in feature selection. *Pattern Recognition Letters*, Elsevier, v. 15, n. 11, p. 1119–1125, 1994. ISSN 0167-8655. Cited in page 41.

QUINLAN, J. R. Bagging, boosting, and c4. 5. In: *AAAI/IAAI, Vol. 1*. [S.l.: s.n.], 1996. p. 725–730. Cited 2 times in pages 42 and 99.

RAGULSKIS, K.; YURKAUSKAS, A. *Vibration of Bearing*. [S.l.]: Hemisphere, 1989. Cited in page 49.

RASCHKA, S. *Python machine learning*. [S.l.]: Packt Publishing Ltd, 2015. Cited in page 26.

RAUBER, T. W.; BOLDT, F. de A.; VAREJÃO, F. M. Heterogeneous feature models and feature selection applied to bearing fault diagnosis. *Industrial Electronics, IEEE Transactions on*, IEEE, v. 62, n. 1, p. 637–646, 2015. Cited 16 times in pages 15, 17, 19, 25, 32, 38, 45, 46, 47, 48, 59, 60, 65, 66, 91, and 101.

RAUBER, T. W.; BOLDT, F. de A.; VAREJÃO, F. M.; RIBEIRO, M. P. Computational intelligence for automatic diagnosis of submersible motor pump conditions in offshore oil exploration. In: *Electronics, Circuits and Systems (ICECS), 2013 20th IEEE International Conference on*. [S.l.: s.n.], 2013. Cited in page 20.

RAUBER, T. W.; BOLDT, F. de A.; VAREJÃO, F. M.; RIBEIRO, M. P. Feature models and condition visualization for rotating machinery fault diagnosis. In: IEEE. *Electronics, Circuits, and Systems (ICECS), 2013 IEEE 20th International Conference on*. [S.l.], 2013. p. 265–268. Cited in page 20.

RAUBER, T. W.; OLIVEIRA-SANTOS, T.; BOLDT, F. de A.; RODRIGUES, A.; VAREJÃO, F. M.; RIBEIRO, M. P. Kernel and random extreme learning machine applied to submersible motor pump fault diagnosis. In: IEEE. *Neural Networks (IJCNN), 2017 International Joint Conference on*. [S.l.], 2017. Cited 3 times in pages 15, 19, and 34.

RIEGER, N. F.; CROFOOT, J. F. *Vibration of Rotationary Machinary*. [S.l.]: Vibration Institute, 1977. Cited in page 49.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, v. 65, n. 6, p. 386–408, 1958. Cited in page 25.

ROUHI, A.; NEZAMABADI-POUR, H. A hybrid method for dimensionality reduction in microarray data based on advanced binary ant colony algorithm. In: IEEE. *2016 1st Conference on Swarm Intelligence and Evolutionary Computation (CSIEC)*. [S.l.], 2016. p. 70–75. Cited in page 41.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. *Learning internal representations by error propagation*. [S.l.], 1985. Cited in page 27.

SMOLENSKY, P.; MOZER, M. C.; RUMELHART, D. E. *Mathematical perspectives on neural networks*. [S.l.]: Psychology Press, 2013. Cited in page 27.

SOKOLOVA, M.; LAPALME, G. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, Elsevier, v. 45, n. 4, p. 427–437, 2009. Cited 5 times in pages 32, 34, 35, 78, and 87.

STAŃCZYK, U. Ranking of characteristic features in combined wrapper approaches to selection. *Neural Computing and Applications*, Springer, v. 26, n. 2, p. 329–344, 2015. Cited in page 39.

Support vector machine. *Support vector machine — Wikipedia, The Free Encyclopedia*. 2017. [Online; accessed 31-March-2017]. Available from Internet: <https://en.wikipedia-.org/wiki/Support_vector_machine>. Cited in page 29.

TAKÁCS, G. *Electrical Submersible Pumps Manual: Design, Operations, and Maintenance*. [S.l.]: Elsevier, 2009. Cited in page 57.

TAPSON, J.; SCHAIK, A. van. Learning the pseudoinverse solution to network weights. *Neural Networks*, Elsevier, v. 45, p. 94–100, 2013. Cited in page 30.

TAVNER, P. J.; RAN, L.; PENMAN, J.; SEDDING, H. *Conditiong Monitoring of Rotating Electrical Machines*. London: The Institution of Engineering and Technology, 2008. ISBN 978-0863417399. Cited in page 57.

TSAI, C.-F.; EBERLE, W.; CHU, C.-Y. Genetic algorithms in feature and instance selection. *Knowledge-Based Systems*, Elsevier, v. 39, p. 240–247, 2013. Cited in page 41.

VAPNIK, V. *The nature of statistical learning theory*. New York: Springer-Verlag, 1999. Cited in page 28.

VAPNIK, V. N.; KOTZ, S. *Estimation of dependences based on empirical data*. [S.l.]: Springer-Verlag New York, 1982. Cited in page 28.

VIOLA, P.; JONES, M. Rapid object detection using a boosted cascade of simple features. In: IEEE. *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*. [S.l.], 2001. v. 1, p. I–511. Cited 2 times in pages 17 and 77.

VONG, C.-M.; WONG, P.-K.; IP, W.-F. A new framework of simultaneous-fault diagnosis using pairwise probabilistic multi-label classification for time-dependent patterns. *IEEE Transactions on Industrial Electronics*, v. 60, n. 8, p. 3372–3385, Aug 2013. ISSN 0278-0046. Cited in page 69.

WANDEKOKEM, E. D.; MENDEL, E.; FABRIS, F.; VALENTIM, M.; BATISTA, R. J.; Varejão, F. M.; RAUBER, T. W. Diagnosing multiple faults in oil rig motor pumps using Support Vector Machine classifier ensembles. *Integrated Computer-Aided Engineering*, IOS Press, v. 18, n. 1, p. 61–74, 2011. Cited 3 times in pages 16, 43, and 58.

WANDEKOKEN, E. D.; VAREJAO, F. M.; BATISTA, R.; RAUBER, T. W. Support vector machine ensemble based on feature and hyperparameter variation for real-world machine fault diagnosis. In: *Soft Computing in Industrial Applications*. [S.l.]: Springer, 2011. p. 271–282. Cited 4 times in pages 16, 17, 29, and 42.

WASHINGTON, U. of. *Tennessee Eastman Challenge Archive*. 2015. Http://depts.washington.edu/control/LARRY/TE/download.html. Accessed: 2016-06-08. Cited in page 52.

WEI, M.; CHOW, T. W.; CHAN, R. H. Heterogeneous feature subset selection using mutual information-based feature transformation. *Neurocomputing*, Elsevier, v. 168, p. 706–718, 2015. Cited in page 17.

WEI, Z.; GAO, J.; ZHONG, X.; JIANG, Z.; MA, B. Incipient fault diagnosis of rolling element bearing based on wavelet packet transform and energy operator. *WSEAS TRANSACTIONS on SYSTEMS*, World Scientific and Engineering Academy and Society (WSEAS), v. 10, n. 3, p. 81–90, 2011. Cited in page 49.

WERBOS, P. J. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. Tese (Doutorado) — Harvard University, 1974. Cited in page 27.

WIDROW, B.; HOFF, M. E. *Adaptive Switching Circuits*. Stanford, California, 1960. Cited 2 times in pages 25 and 27.

WIDROW, B.; LEHR, M. A. 30 years of adaptive neural networks: perceptron, Madaline, and backpropagation. *Proceedings of the IEEE*, v. 78, n. 9, p. 1415–1442, ago. 2002. Available from Internet: <http://dx.doi.org/10.1109/5.58323>. Cited 2 times in pages 25 and 26.

WILLIAMS, D.; HINTON, G. Learning representations by back-propagating errors. *Nature*, v. 323, n. 6088, p. 533–538, 1986. Cited in page 27.

WU, S.-D.; WU, P.-H.; WU, C.-W.; DING, J.-J.; WANG, C.-C. Bearing fault diagnosis based on multiscale permutation entropy and support vector machine. *Entropy*, Molecular Diversity Preservation International, v. 14, n. 8, p. 1343–1356, 2012. Cited 2 times in pages 45 and 46.

XIA, Z.; XIA, S.; WAN, L.; CAI, S. Spectral regression based fault feature extraction for bearing accelerometer sensor signals. *Sensors*, v. 12, n. 10, p. 13694–13719, 2012. ISSN 1424-8220. Available from Internet: <http://www.mdpi.com/1424-8220/12/10/13694>. Cited 4 times in pages 45, 46, 47, and 49.

XU, Y.; CHEN, Y. J.; ZHU, Q. X. An extension sample classification-based extreme learning machine ensemble method for process fault diagnosis. *Chemical Engineering & Technology*, Wiley Online Library, v. 37, n. 6, p. 911–918, 2014. Cited in page 52.

XU, Y.; DENG, X. Fault detection of multimode non-gaussian dynamic process using dynamic bayesian independent component analysis. *Neurocomputing*, Elsevier, v. 200, p. 70–79, 2016. Cited in page 52.

YIN, S.; DING, S. X.; HAGHANI, A.; HAO, H.; ZHANG, P. A comparison study of basic data-driven fault diagnosis and process monitoring methods on the benchmark Tennessee Eastman process. *Journal of Process Control*, v. 22, n. 9, p. 1567 – 1581, 2012. ISSN 0959-1524. Cited 3 times in pages 15, 52, and 69.

YIN, S.; LUO, H.; DING, S. Real-time implementation of fault-tolerant control systems with performance optimization. *Industrial Electronics, IEEE Transactions on*, v. 61, n. 5, p. 2402–2411, May 2014. ISSN 0278-0046. Cited 2 times in pages 15 and 52.

YONG, Z.; DUN-WEI, G.; WAN-QIU, Z. Feature selection of unreliable data using an improved multi-objective {PSO} algorithm. *Neurocomputing*, v. 171, p. 1281 – 1290, 2016. ISSN 0925-2312. Cited in page 41.