FAULT DIAGNOSIS SYSTEM OF CSTR PROCESS BASED ON STACKING CLASSIFIER ALGORITHM

Clynton Roger Guastti de Oliveira

Daniel Cruz Cavalieri

Cassius Zanetti Resende

clyntonroger@gmail.com

dcruzcavalieri@gmail.com

cassius@ifes.edu.br

Instituto Federal do Espírito Santo

ES-010, Km-6,5 - Manguinhos, 29173-087, Serra, Espírito Santo, Brasil

Abstract. *In recent years, fault diagnosis in industrial processes has been widely studied. Various methods based in artificial neural networks and statistical classifiers have been used to extract the root causes of faults. This paper presents a classification performance comparison between a Support Vector Machine (SVM) classifier and an Artificial Neural Network (ANN) applied to solve a multiclass fault diagnosis problem. A chemical process simulator containing a Continuous Stirred Tank Reactor (CSTR) provides the analyzed syntactic data with linear and nonlinear faults. Additionally, a stacking algorithm is proposed to improve the classification accuracy by combining the two classifiers. Finally, a comparison between the three classifiers and their performance for classifying ten types of faults is presented, showing the improvement of the classification with the use of the stacking algorithm proposed.*

Keywords: *Support Vector Machine, Artificial Neural Network, Continuous Stirred Tank Reactor, Stacking Algorithm, Multiclass Classifiers*

1 INTRODUCTION

The increasing demands for low-cost and competitive industrial production have stimulated the search for techniques that allow greater availability of manufacturing plants. In this context, fault diagnosis methods allow abstraction of the knowledge of maintenance specialists, making possible the reduction of the operational unavailability of the plants.

In the last years, various methods based on artificial inteligence have been used to identify and classify process faults. In Miljković (2011) a list of works of fault diagnosis involving electric motors, pumps, bearings, among other important systems are reported, showing that there is a wide range of industrial applications that can be developed using these methods.

Srihari et al. (2010) used a gearbox vibration signature preprocessed in order to obtain the required features of mean, root mean square (rms), standard deviation, skewness and kurtosis and applied these features as input parameters to the artificial neural network. The high accuracy achieved shows that neural networks can be effectively used to improve the detection of failures in the industrial environment.

Pandya et al. (2010) used Wavelets to extract features of mechanical vibration signals from bearings and applied to a neural network for failure detection. The results showed an accuracy greater than 93%. Other works have explored the combination of classifiers to improve the accuracy of fault detection and pattern recognition systems. André (2013) used a classifier combining k-NN and SVM to detect faults in rotary machines, obtaining an accuracy of 75% to 85% for different machines.

This paper presents a classification performance comparison among a Support Vector Machine (SVM), a k-Nearest Neighbor, a Discriminant Analysis classifiers and an Artificial Neural Network (ANN) applied to solve a multiclass fault diagnosis problem based in process history. The historical dataset was obtained from the CSTR process simulator developed by MIT aiming at simplifying the data preprocessing step, since the focus of the work is the comparison of the classifiers performance. The stacking algorithm is presented as a good option to improve the classification perfomance.

The paper is organized as follows: The basic terminology of fault diagnosis is presented in Section 2. Next, in Section 3, a review of the theory behind Support Vector Machine (SVM), k-Nearest Neigbhor (k-NN), Quadratic Discriminant Analysis (QDA), Artificial Neural Network (ANN) and Stacking Method is presented. Section 4 presents the Continuous Stirred Tank Reactor Simulator. The methodology and experiments are described in Section 5. Section 6 presents the results and arguments, and finally Section 7 presents the conclusion, the summary of contributions and the description for future developments.

2 FAULT DIAGNOSIS

2.1 Basic Terminology

Initially, given the variety of terms and interpretations on the subject, it is necessary to define a basic terminology for understanding the work. Three basic concepts can be defined (Miljković, 2011):

- **Faults** are unauthorized deviations of one or more characteristics of the system when compared to the considered acceptable state.

- **Failures** are permanent interruptions of the system conditions performing their basic functions, under certain operating conditions, usually results from one or more faults.
- **Malfunction** is an intermittent condition that prevents the system from being used in the desired way.

These conditions are identified from the qualitative or quantitative analysis of the set of historical data extracted from the plant. According to Miljković (2011), the evolution of fault results in failures or malfunction of the system, as can be observed in the diagram of Figure 1, which shows the system in normal operation (logic level 1) and its transition to abnormal condition (logic level 0).

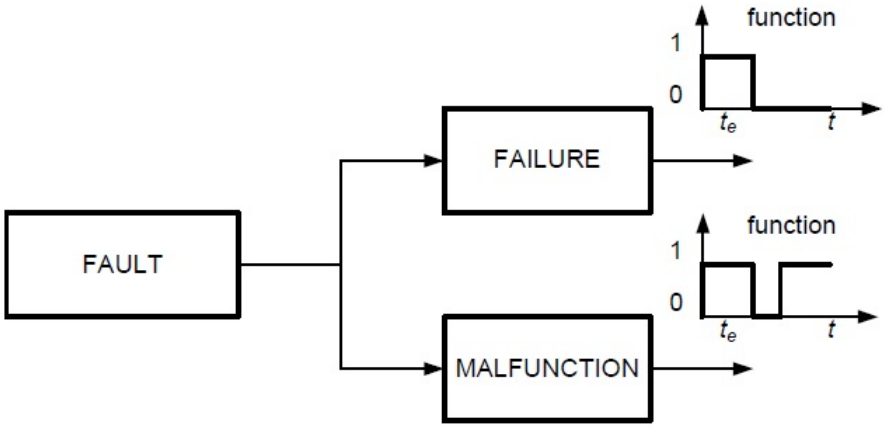

Figure 1: Progression of fault towards failure or malfunction.

The types of faults can be divided based on faulty components (actuator faults, plant component faults and sensor faults), on the faulty form (abrupt, incipient and intermittent faults) or based on the form wich the fault is added to the system (additive or multiplicative). The Figure 2 shows faults based on faulty form (Miljković, 2011).

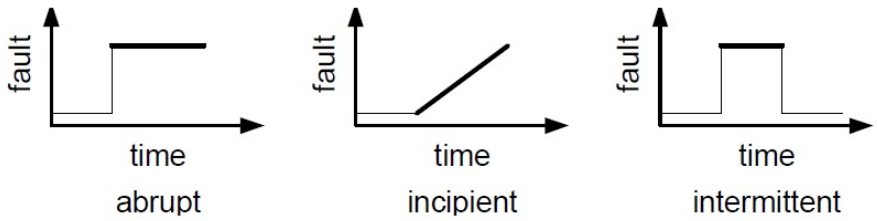

Figure 2: Fault models based on faulty form

Finally, according to Miljković (2011), the diagnosis of failures also involves some concepts to differentiate detection, isolation and identification:

- **Detection** determines that a fault state has occurred.
- **Isolation** determines the location and type of fault.
- **Identification** determines the magnitude of the fault.

2.2 Fault Detection Methods

The fault detection system is based on two basic components: the type of knowledge and the diagnostic search strategy. The knowledge can be referred in a deep, causal or mathematical process model, it may also be based on past experience with the process. This knowledge is referred to a shallow, compiled, evidential or process history-based knowledge (Venkatasubramanian et al., 2003).

The second component is the type of diagnostic search strategy. This strategy is a knowledge representation which presents the relationship between the observations (symptoms or features) and the failures. It can be classified as qualitative or quantitative model. According to Venkatasubramanian et al. (2003), the fault diagnostic methods can be separated into three large groups as shown in the diagram of Figure 3. In this work, diagnostic methods based on process history data and quantitative models has been used applied to a CSTR simulator.

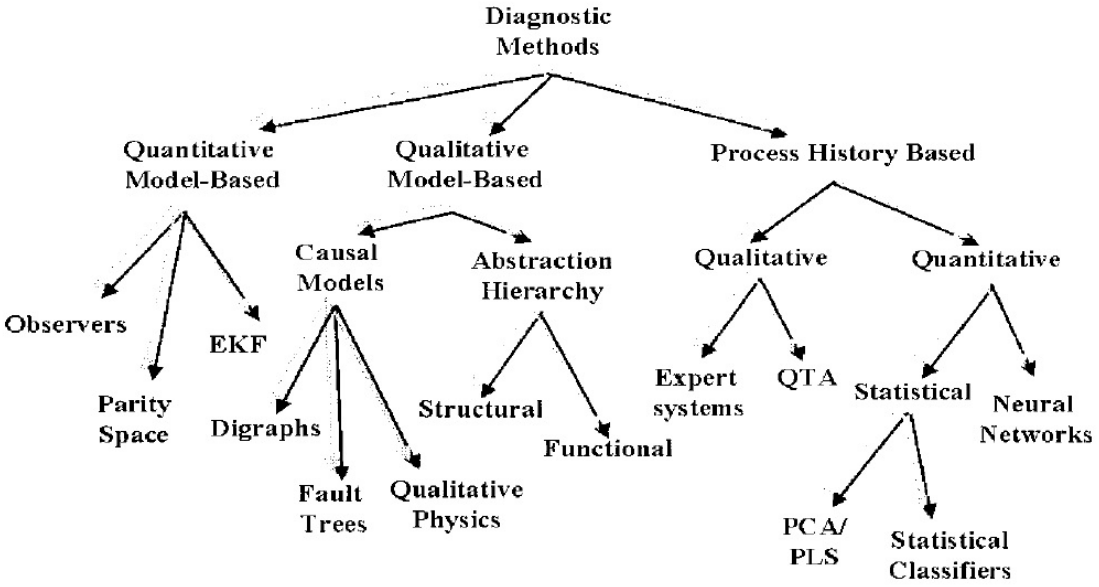

Figure 3: Fault models based on faulty form

3 CLASSIFIERS

The classifiers label the fault according to a previously established standard. Several types of quantitative classifiers are available and each method is often more effective with certain datasets. For this reason, studies seeking the combination of classifiers have multiplied, aiming at the best characteristics of each classifier in the process of data analysis.

3.1 k-Nearest Neighbors algorithm

The k-nearest neighbor classifier is a very simple and efficient technique. It is widely used in many areas such as pattern recognition, object recognition, text analysis and general recognition applications. Basically, the nearest neighbor is calculated on the basis of value of k, and it defines how many nearest neighbors are be used to define the class of each sample (Bhatia, 2010).

The distance is calculated based on a metric and the sample class can be defined as the one with the highest number of representatives among the K points identified. In this work the distance is calculated by Euclidean distance.

According to Fukunaga (2013) and Matworks (2010), when the Euclidean distance is used, given an mx-by-n data matrix X, which is treated as mx (1-by-n) row vectors $x_1, x_2, ..., x_{mx}$, and my-by-n data matrix Y, which is composed of my (1-by-n) row vectors $y_1, y_2, ..., y_{my}$, the distances between the vector X_s and y_t are defined as shown in Eq. (1).

$$d_{st}^2 = (x_s - y_t)(x_s - y_t)' \tag{1}$$

3.2 Discriminant Analysis

Discriminant Analysis assumes that different classes generate different parameters of the Gaussian distributions. The decision theory for classification shows that is necessary to know the class posteriors Pr(G—X) for optimal classification. Suppose fk(x) is the class-conditional density of X in class G = k, and let πk be the prior probability of class k, with PK k=1 πk = 1. Equation (2) shows a simple application of Bayes theorem (Hastie et al., 2009).

$$Pr(G = k|X = x) = \frac{f_k(x)\pi_k}{\sum_{l=1}^{K} f_l(x)\pi_l} \tag{2}$$

It can be seen that in terms of ability to classify, having fk (x) is almost equivalent to having the quantity Pr(G=k—X=x). The linear discriminant analysis (LDA) and quadratic discriminant analysis (QDA) use Gaussian densities for data classification. In this work the QDA classifier has been used with a pseudoquadratic Matlab function to eliminate the discriminant's fails.

3.3 Support Vector Machine

Support Vector Machine (SVM) is originally a binary classification method. Basically, the SVM proposes that two classes can be separated by a hyperplane, as shown in Figure 4 (Hastie et al., 2009).

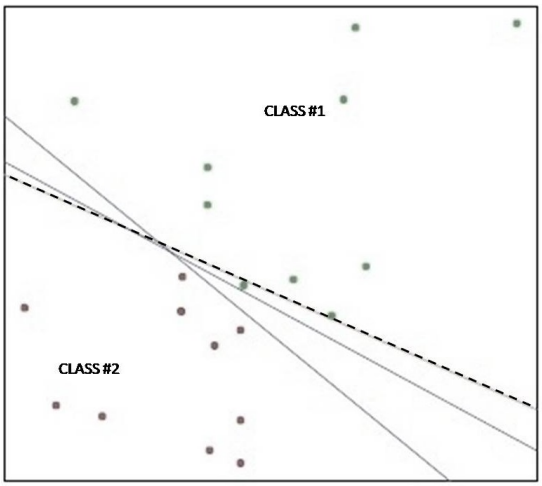

Figure 4: Separating hyperplanes

The two solid lines are two of the infinitely many possible separating hyperplanes. The dashed line is the least squares solution to the problem, obtained by regressing the 1/1 response Y on X (with intercept); the line is given by Eq.(3).

$$x : \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 = 0 \tag{3}$$

Kernel functions can be used to simplify the feature extraction from large datasets that can be very costly or even unfeasible. In this work the Gaussian kernel function was adopted.

3.4 Artificial Neural Networks

Artificial neural networks are computational models developed as the human brain. They are represented by a set of processing units, called artificial neurons, interlinked by interconnections (synapses) implemented by vectors and matrices of synaptic weigths. These artificial neurons, inspired by the analysis of human neuron, are nonlinear, usually providing continuous outputs and based in simple functions as in Figure 5 (Silva et al., 2016).

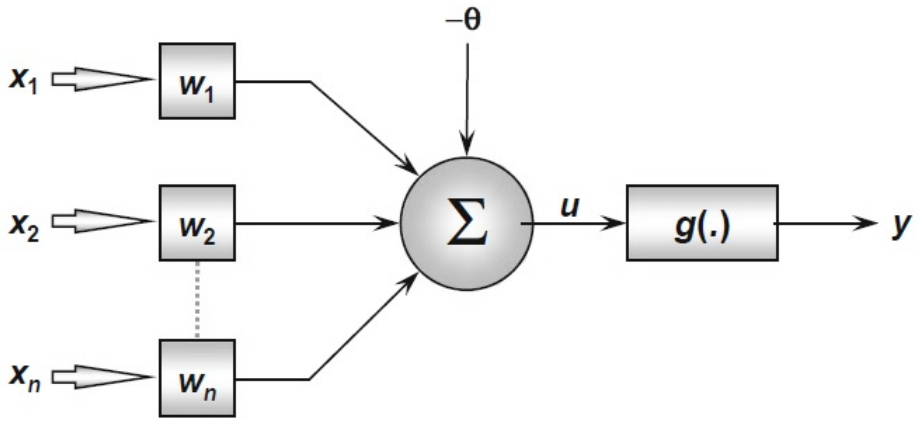

Figure 5: The artificial neuron.

Basically, each neuron in a network can be built as shown in Figure 5, where multiple input signals arrive from the external environment and are represented by the set $x_1, x_2, x_3, ..., x_n$. In it the so-called synaptic weights $w_1, w_2, w_3, ..., w_n$ can be observed and the relevance of each of the x_i neuron inputs is calculated by multiplying them by their corresponding synaptic weight w_i. Therefore, the output of the neuron, denoted by u, is the weighted sum of its inputs.

The Figure 5 also shows the linear aggregator that gathers all input signals weigthed to produce an activation, the bias (θ) that is used to specify the proper threshold that generate a trigger value in the neuron output and the activation function, which limits the output of the Neuron within a reasonable range of values.

The Eq. (4) and Eq. (5) synthesize the result produced by the artificial neuron proposed by McCulloch and Pitts. Different activation functions can be used to improve neural network performance.

$$u = \sum_{i=1}^{n} w_i . x_i - \theta \tag{4}$$

$$y = g(u) \tag{5}$$

According to Silva et al. (2016), an ANN can be divided into three parts, named layers:

- **Input layer** - This layer receives information from the external environment. These signals are features or measurements.
- **Hidden, intermediate, or invisible layers** - They are responsible for features extraction from the process.
- **Output layer** - This layer is responsible for producing and presenting the final network outputs based on the result of the neurons processing in the previous layers.

When the ANN is built, several neurons are associated and the number of hidden layers and their respective amount of neurons depends on the nature and complexity of the problem, as well as the quantity and quality of the available data about the problem. The Figure 6 shows a feedforward network architecture with two hidden layers (Silva et al., 2016).

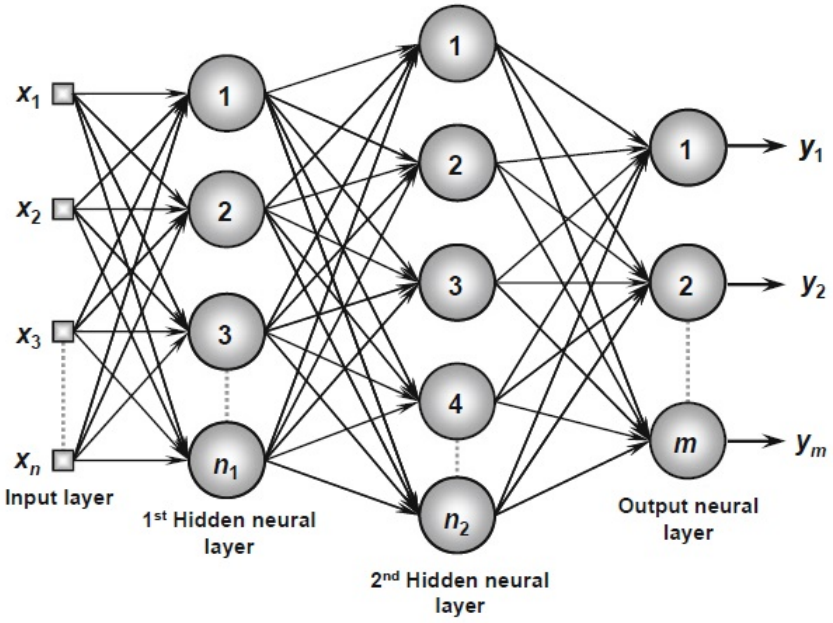

Figure 6: Example of a feedforward network with multiple layers.

In this work an ANN is used with two hidden layers, one with ten and other with three neurons respectively.

3.5 Stacking

The Stacking method can be considered as a meta-learning approach based on two levels of classifiers: the first one, composed of more complex classifiers, receives the set of attributes of the external environment and performs a first classification. The second level, usually composed of a simpler classifier, is learnt to combine the firt-level classifiers output (Tang, 2015).

In this work, the first level of classifiers is composed of a Gaussian SVM classifier, a K-NN classifier and a QDA. In the second level a Gaussian SVM classifier was used. To improve the classification, in each level a standardization of the variables was done by subtracting each sample by the mean and dividing by the standard deviation. The basic algorithm can be described as follows (Tang, 2015):

#	**Stacking algorithm**
Input:	Training data $D = x_i, y_{i i=1}^{m}(x_i, y_i)$
Output:	An ensemble classifier H
Step 1:	Learn first-level classifiers
	for t 1 to T **do**
	Learn a base classifier h_t based on D
	end for
Step 2:	Construct new data sets from D
	for i 1 to m **do**
	Construct a new data set that contains x_i^l, y_i, where $x_i^l = h_1(x_i), h_2(x_i), ..., h_T(x_i)$
	end for
Step 3:	Learn a second-level classifier
	Learn a new classifier h' based on the newly constructed data set
	return $H(x) = h'(h_1(x), h_2(x), ..., h_T(x))$

4 CONTINUOUS STIRRED TANK REACTOR

Process simulators are often used for evaluation of fault-diagnosis systems. In this work a Continuous Stirred Tank Reactor (CSTR) based on two doctoral theses from the Massachusetts Institute of Technology is used to generate the database for analysis. This model is considered benchmark and is used in simulations in several areas such as chemical, hydraulic, mechanical and automation engineering.

In this process, two globally exothermic competing reactions are transforming reactant A into products B and C. In general, the process works as follows:

- The reactant A with concentration C_{A0} is entering the reactor with flow rate Q1 and temperature T1.
- The products and rest of reactant are pumped out of tank, with flow rate Q2 wich is divided between flow rate Q4 (effluent pipe) and Q3 (eventual leak).
- L is the level and it is controlled by a valve as an actuator.
- The jacket arround the tank is responsible for keeping its temperature T2 constant.
- The cascade controller varies the inflow of the cooling liquid at flow rate Q5 and temperature T3.
- Leaks from jacket to the exterior at flow rate Q7 and from the jacket into the CSTR at flow Q6 can be introduced into the simulation faults.

The Figure 7 shows the process flow diagram of the simulator.

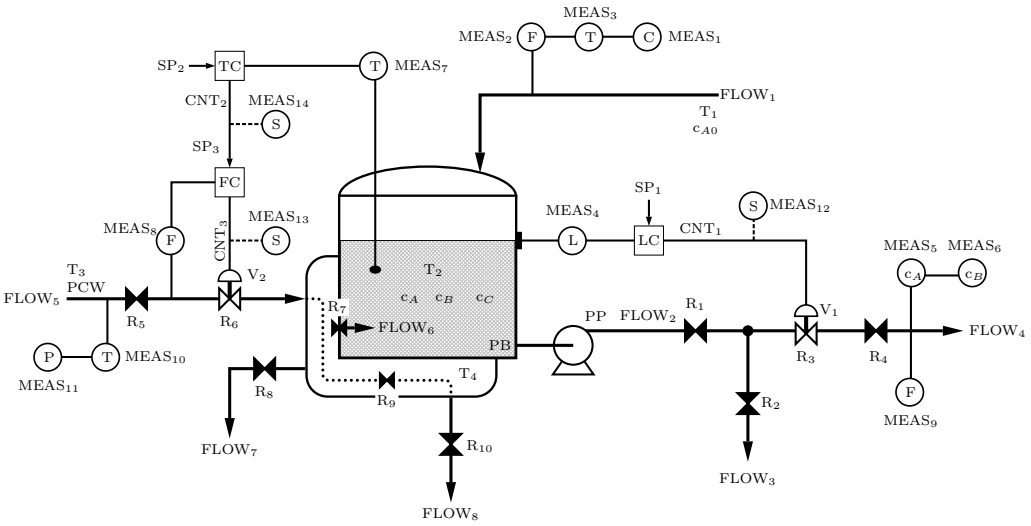

Figure 7: Continuous Stirred Tank Reactor
Source: https://sites.google.com/site/cstrsimulator

The simulator allows the simulation of several types of linear and non-linear faults. The Table 1 shows a total of 14 measure variables used as features for fault diagnosis, as well as their nominal values and units of measurement.

Table 1: Measure variables and constraints of the CSTR simulator

#	Variable/Constraint	Acronym	Nominal	Units
1	Feed concentration	C_{A0}	20.0	mol/m^3
2	Feed flowrate	Q1	0.25	m^3/s
3	Feed temperature	T1	30.0	C
4	Reactor level	L	2.0	m
5	Product A concentration	C_A	2.85	mol/m^3
6	Product B concentration	C_B	17.114	mol/m^3
7	Reactor temperature	T2	80.0	C
8	Coolant flowrate	Q5	0.9	m^3/s
9	Product flowrate	Q5	0.9	m^3/s
10	Coolant inlet temperature	T3	20.0	C
11	Coolant inlet head	h7	10.0	m
12	Level controller output	m1	10.16	%
13	Coolant flow controller output	m2	61.0	%
14	Coolant flow controller setpoint	m2	0.907	%

In addition, four constraints are calculated as additional features. A fixed seed for the random number generator is provided in the simulation configuration to allow the reproducibility of the experiments. The fault simulation is realized by changing some of the internal system parameters. The Table 2 shows the possible process faults available for simulation and the respective parameter that is changed in the model.

Table 2: Process faults

#	Fault Type	Affected Parameter	Nominal Value
1	No fault	-	-
2	Blockage at tank outlet	K1	10.0(10.0,300.0)
3	Blockage in jacket	K9	0.0(0.0,1.0]
4	Jacket leak to environment	m_8^L	0.0(0.0,1.0]
5	Jacket leak to tank	m8, K8	0.0(0.0,1.0]
6	Leak from pump	m7, K7	0.0(0.0,1.0]
7	Loss of pump head	m2, K2	0.0(0.0,1.0]
8	Jacket exchange surface fouling	UA_h	1901.0[1600.0,1901.0)
9	External heat sink/source x 1000	qext	0[-10,10]
10	Primary reaction activation	EB	25(25,30]
11	Secondary reaction activation	EC	45(45,54]
12	Abnormal feed flowrate	Q1	0.25[0.0,0.35]
13	Abnormal feed temperature	T1	30.0[10.0,50.0]
14	Abnormal feed concentration	C_{A0}	20.0[0.0,30.0]
15	Abnormal cooling water temperature	T3	20.0[0.0,40.0]
16	Abnormal cooling water head	h7	10.0[0.0,15.0]
17	Abnormal jacket effluent head	h10	1.4[-100.0,100.0]
18	Abnormal reactor effluent head	h1	26.0[-200.0,200.0
19	Abnormal level controller SP	Z4	2.0[1.5,2.5]
20	Abnormal temperature controller SP	r3	80.0[70.0,90.0]
21	Control valve 1 stuck	m1	0.1059[0.0,1.0]
22	Control valve 2 stuck	m2	0.9[0.0,1.0]

5 METHODOLOGY

Initially, the database was created from the CSTR simulator. The simulation scenario consists of seven different faults (12, 13, 14, 15, 16, 21 and 22) and the normal condition (class 1).

For each failure, 10 simulations were performed with 800 samples and different random Seeds {17, 23, 327 , 53, 222, 3789, 98, 7, 31, 457}. The simulation time was 1249 minutes with Δt= 1.0 min and the fault delay set to 0 min. Finally, after the simulations, a single 64000 x 18 data matrix was formed.

The database was divided into 70% of data for training and 30% for testing and validation of the system. For training of the Stacking classifier the training base (70% Dataset1) was again divided, using 70% of the samples in the training of the first level classifiers and 30% in the training of the meta-classifier.

The holdout function was used for the task of dividing the database at each iteration, in order to ensure that different data were used for each training and validation cycle. The Figure 8 shows the flowchart used in this work.

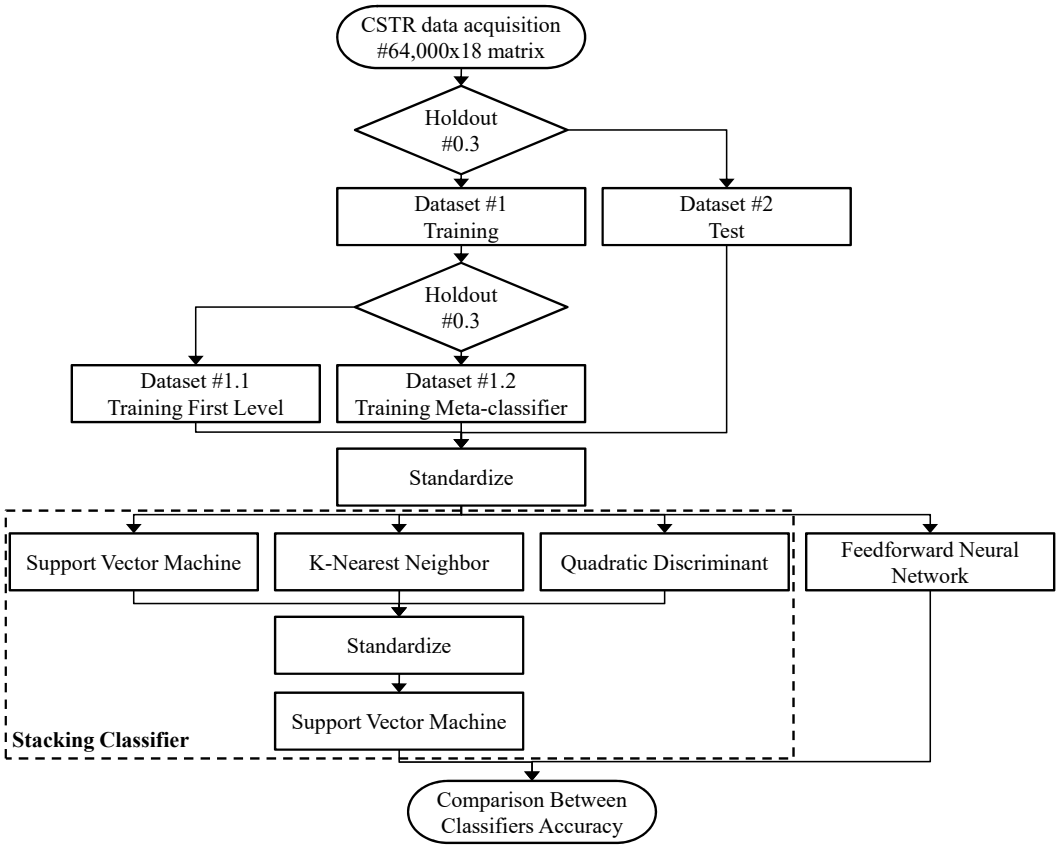

Figure 8: The Stacking Flowchart.

At first the data obtained from the simulation were used without the application of any type of filter or pre-treatment. The accuracy of the SVM classifier (SVM, k-NN and QDA) ranged from 42% to 55% for the SVM classifier, with a final classification in the meta-classifier between 46% and 58%. Those indexes are unsatisfactory. This result pointed to the necessity of conditioning the databases shown in Figure 8 in the ”Standardize” step.

The features were normalized in the step ”Standardize” by subtracting each sample by the mean and divided by the standard deviation. This process was done in both the input attributes of the first-level classifiers and the meta-class input data (first-level output).

The classifiers shown in Figure 8 were implemented using the template classifiers functions from MATLAB Statistics Toolbox. Some configurations used are listed below, in order to allow the reproducibility of the experiments:

- **General Note** - All statistical classifiers were trained using the one-versus-one coding design and Standardize function in 'true'. The optimization parameters for the classifiers were chosen through iterative processes based on the lowest classification errors.
- **Support Vector Machine** - The Gaussian kernel was used with KernelScale parameter in 'auto'. When KernelScale is 'auto', the software selects an appropriate scale factor using a heuristic procedure, and divides all elements of the predictor matrix X by the value of KernelScale. Then, the software applies the appropriate kernel norm to compute the Gram matrix.
- **k-Nearest Neighbors** - A K value of 7 and the Euclidean distance were used.
- **Quadratic Discriminant Analysis** - The pseudoquadratic funtion was used to eliminate the processing failures.
- **Feedforward Neural Network** - It was used with two hidden layers, one with ten and the other with three neurons with Matlab default parameters.

For the comparison between the classifiers, 50 experiments were performed and the mean and standard deviation of the acuracies of each structure were calculated. Accuracy was calculated from the comparison of the predicted labels with the original base.

The meta-classifier is composed exactly by the same SVM classifier used in the first level, aiming to demonstrate the improvement in the classification provided by the combination of classifiers. However, it is important to say that the Naive Bayes and k-NN algorithms were initially tested in the meta-classifier function, as suggested by the literature (Dzeroski & Zenko 2004), and presented lower results when the entire database was used, with an average accuracy of 92 % and 95.3 %, respectively.

6 RESULTS AND ARGUMENTS

The Table 3 shows the best and worst case (the highest and the lowest accuracy), the mean and the standard deviation reached after 50 classification cycles. It is possible to observe that, although the average precision of the stacking algorithm was superior to others, the SVM classifier presented the best standard deviation, that is, reproducibility of results. In contrast, it also presented the second worst average accuracy, just after the feedforward neural network.

It is important to note that the worst performance of Stacking (92.1719%) was not obtained from a combination of bad results from the first level classifiers (SVM = 92.1927%, k-NN = 91.7292% and QDA = 92.0417%), but from the combination of the classes that the classifiers mistakenly identified. That is, the success of the Stacking Algorithm is directly related to which classes the first level classifiers can classify correctly, being important that they do not make similar errors. The use of the same SVM classifier in the second level allows the potential for improvement that the Stacking Method can offer to the fault detection system.

Table 3: Results of experiments based on 50 cicles [%].

	SVM	k-NN	QDA	ANN	Stacking
Best Case	92.7500	99.5781	99.7917	89.2634	99.8958
Worst Case	92.0260	91.6458	91.8438	18.6905	92.1719
Mean	92.3662	92.4389	92.6676	84.3676	98.9146
Standard Deviation	0.1646	1.8187	1.8136	18.3817	2.4669

Further studies can be carried out aiming at reducing the variability of the results based on the fixation of the random seed in a pre-established value or even applying classifiers of lesser complexity in the meta-classifier function.

Another line of research that can be explored as a way to improve the performance of the structure is to perform an analysis of the classes identified in level 1 by the method of Sensitivity and specificity or even of matrices of confusion or ROC curves. The complete identification of the classes with the highest error rate may allow an optimization in the first level classifiers to improve classification in the level 0.

Finally, the fact that this study was developed from data generated in a process simulator opens a discussion about what will be the data conditioning required when real data are used, because in the present, problems like outliers, sampling and different production levels have been suppressed from simulations.

7 CONCLUSION

In this work, the evaluation of the Stacking in the fault detection was made from the comparison with four other classifiers, corroborating in showing the potential of this method for the improvement of the systems of automatic detection of faults.

Although the literature recommends the use of simpler classifiers in the meta-classifier function (Dzeroski & Zenko 2004), the results point in the same direction as Caffé et al. (2011) that in the application of Stacking some problems may require meta-classifiers with a greater degree of complexity.

In general, the data pre-treatment used and the Stacking algorithm were promising in the detection of simulated faults in the CSTR process simulator. Future studies can also evaluate the behavior of the proposed structure with real data, which should require greater effort in the feature extraction step.

REFERENCES

Andre, A. B., Beltrame, E., Wainer, J., 2013. A combination of support vector machine and k-nearest neighbors for machine fault detection. *Applied Artificial Intelligence*, vol. 27, n.1, pp. 36-49.

Bhatia, N., 2010. Survey of nearest neighbor techniques. In *(IJCSIS) International Journal of Computer Science and Information Security*, vol. 8, n.2, pp.302-305.

Burges, C. J., 1998. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, vol. 2, n.2, pp. 121-167.

Caffé, M. I. R., Perez, P. S., Baranauskas, J. A., 2011. Avaliação do Algoritmo de Stacking em Dados Biomédicos. In *Anais do XXXI Congresso da Sociedade Brasileira de Computação, XI Workshop de Informática Médica*, Natal, RN.

Dzeroski, S., Zenko, B., 2004. Is combining classifiers with stacking better than selecting the best one?. In *Machine learning*, vol. 54, n.3, pp. 255-273.

Fukunaga, K., 2013. *Introduction to statistical pattern recognition*. Academic press.

Hastie, T., Tibshirani, R., Friedman, J., 2009. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction Second Edition*. Biometrics.

Loh, W. Y., 2011. Classification and regression trees. Wiley Interdisciplinary. *Reviews: Data Mining and Knowledge Discovery*, vol. 1, n.1, pp. 14-23.

Madzarov, G., Gjorgjevikj, D., & Chorbev, I., 2009. A multi-class SVM classifier utilizing binary decision tree. *Informatica*, vol. 33, n.2, pp. 233-241.

Matworks, 2011. *Statistics Toolbox User's Guide 2011b*. The MathWorks Inc. Natick, Massachusetts, United States.

Miljkovi, D., 2011. Fault detection methods: A literature survey. In MIPRO, *proceedings of the 34th international convention IEEE.*, pp. 750755.

Silva, I. N., Spatti, D. H., Flauzino, R. A., Liboni, L. H. B., dos Reis Alves, S. F., 2016. *Artificial Neural Networks: A Practical Course*. Springer.

Tang, J., S. Alelyani, and H. Liu, 2015. *Data Classification: Algorithms and Applications. Data Mining and Knowledge Discovery Series*. CRC Press, pp. 498-500.

Venkatasubramanian, V., Rengaswamy, R., Yin, K., Kavuri, S. N., 2003. A review of process fault detection and diagnosis: Part I: Quantitative model-based methods. In Computers chemical engineering, vol. 27, n.3, pp. 293-311.