Streamlined Machine Learning Method for Industry 4.0 and Embedded Applications

Leandro Landgraf
Faculty of Engineering, Environment and Computing
Coventry University
Coventry, United Kingdom
landgrafl@uni.coventry.ac.uk

***Abstract* — With the advent of Industry 4.0 and the Internet of Things (IoT), timely and precise monitoring and control of processes and sensors are essential to ensure high quality, reduce disruptions and waste of time and money. In this paper, a new machine learning method that can be equally suitable for those uses is proposed and demonstrated. This method employs a moving window approach in the features engineering step and weighted Radial Basis Function Support Vector Machines (RBF-SVM) for modelling and detection, and is proven to be both efficient and flexible for a broad range of applications.**

Keywords — time series, classification, support vector machines, moving window, pattern recognition, automation, condition monitoring

I. Introduction and Related Work

For many years, the use of statistical process control (SPC) techniques has been a key factor in the continuous improvement of both the manufacturing processes and the quality of the industrial products, and the most important and frequently used tools are the control chart patterns (CCPs). Any deviations in men, machine, materials, methods, measurements or the environment (5M1E) will reflect in the chart, in the form of many different patterns, of which six are present in this study: normal, cyclic, increasing and decreasing trends, and upward and downward shifts. Prompt and early identification of abnormal patterns is critical to ensure corrective actions are taken timely and effectively.

The initial methods used for chart identification were the basic Euclidean distance calculation, a method that is not precise and prone to misidentifications because similar sequences can be separated by a large distance [4], and several rule-based ones that could neither properly describe the abnormal patterns nor avoid many false alarms that would require further manual inspection and expert knowledge.

The traditional methods have long been replaced by machine learning ones, including self-organising neural networks [15] in the early days, and more recent approaches that use features extracted from the raw data and calculations from the wavelet, shape and statistical domains [24] together with several different machine learning methods like support vector machines (SVM) using multi-kernel for kernel function combination [11], combined supervised locally linear embedding (SLLE) with SVM [26] or artificial neural networks (ANN) with 30 different shape features [7]. Besides methods that rely on an initial features engineering step, some use the raw data and machine learning approaches for both extraction and identification of features, for example, one-dimensional convolutional neural networks (CNN) by [10], [25] and [23], CNNs with Harris hawks optimization [8] or multilayer bidirectional long short-term memory network [24].

Nowadays, with the rise of Industry 4.0 and the Internet of Things (IoT), accurate and precise monitoring and supervision of processes and sensors are essential. The use of Artificial intelligence (AI) and Machine Learning methods have become paramount in achieving early detection of problems, or even in predicting failures, by constant monitoring and assessing CCPs and sensors time-based data. The similarity between CCPs and sensors data is that both are treated as time-series information, and as such, the pre-processing and learning steps are similar for them. To be used with traditional machine learning methods, time-series data is either used in its raw form directly or undergoes a step of feature extraction. As raw data, the time series is usually modified to a tabular format where each data point is treated as a feature on its own, or observations are aggregated in bins like a histogram [24]. The second usual approach is to extract features from the time series and use those features as the attributes, in a tabular format. The problem with those methods is that the time order is lost in the former, and the latter pushes the modelling to be done in the domain of the features, instead of the time domain [18]. Most of the studies reviewed above use one form or the other.

In this paper, similar to the study by [11], we propose a moving window method to calculate the features and a weighted classifier in the same way as [22]. With this method, we intend to channel both benefits of using the raw data and the features extraction approaches, without losing the time references, and additionally account for data class imbalance. As an extra advantage, the method proposed in this study is meant to be used for both CCPs recognition and sensor data applications, and as such, we used two different datasets, described in Part II, for modelling and testing: the univariate data from synthetic CCPs and an example of multivariate data from actual sensors. In Part III the machine learning methods implemented are shortly reviewed, Part IV describes the experimental methodology used and the results are presented in Part V. Finally, the final discussions and conclusions are in Part VI.

II. Datasets Description

Both datasets used for this study are available in the UC Irvine Machine Learning Repository website [20] and [21].

A. Synthetic Control Chart (SCC)

This dataset contains 600 instances, split into 100 samples for each of six different control chart patterns [21], according to the equations in Appendix A – Table IX. The SCC dataset is completely balanced, each instance has 60 sampling points

(time units) and was obtained by simulation. All the patterns, other than the normal, indicate some process alterations, abnormalities or irregularities that must be addressed [15].

The normal (1 – NOR) pattern variations are purely random but all the others can be related to some process deviations: the cyclic (2 – CYC) are composed of sequential occurrences of peaks and troughs and may be caused by factors like the change of operators or voltage fluctuations; trend patterns (upwards trend, 3 – UT, or downwards trend, 4 – DT) show a continuous increase or decrease in the values, possibly caused by operator fatigue, tool wear or equipment deterioration; finally, the shift patterns (upward shift, 5 – US, or downward shift, 6 – DS) appear when a sudden step-like change happens and are an indication of possible machine failures, operators shift changes or implementation of new materials or methods ([23] and [2]).

B. Condition monitoring of hydraulic systems (HS)

The dataset is a multivariate time-series measurement containing 2205 instances of real and virtual sensors, composing 17 time-series that have been experimentally obtained from a hydraulic system test rig [9]. The attributes are all numeric and continuous, and composed of six values for pressure, one value for motor power, two sensors for volume flow, four temperature sensors, one vibration sensor, plus three additional calculated channels, one for efficiency factor, one for the virtual cooling efficiency and one for the virtual cooling power. All the attributes are listed in Appendix A – Table X, and it's worth noting that different attribute groups have different measurement frequencies, ranging from 1Hz to 100Hz.

In terms of target conditions, all measurements are under four different known faults, resulting in four different multiclass classification problems: cooler condition, valve condition, internal pump leakage and hydraulic accumulator pressure (a fifth class, stable flag, only indicates if the system had stabilised and is not used for classification), whose labels and distribution (in terms of the number of instances) are listed in Appendix A – Table XI. Some of the conditions are unbalanced and would require the use of sampling techniques, and all of the 2205 measurements are 60 seconds long, the same length of the SCCs used.

III. Machine Learning Methods

A. Decision Trees (DT)

Decision Trees are a non-parametric algorithm that works by recursively splitting the data into two groups, also called branches, until no further splits are possible (for example, only one sample is left in the leaf). Due to this process, DTs are prone to overfitting, because it is easy to go deep into the tree and ending with too much focus on the details of individual samples. One way to avoid this is to use the technique of pruning, that is, purposively cutting the tree short based either on the maximum deep or the minimum number of samples before splitting. Another possibility is in using multiple trees, resulting in an ensemble of trees, of which the Random Forest Classifier is an example.

B. Logistic Regression (LR)

Despite the name, Logistic Regression is a linear classification algorithm that can use the cross-entropy loss for multiclass applications without the need of being decomposed into binary classification problems using the One-vs-One or One-vs-Rest techniques[14]. Employing a logistic function (hence the name) the method models the probability for each sample to belong to a certain class, and the choice is made based on a threshold.

C. Support Vector Machines (SVM)

Support Vector Machines used for classification are a category of discriminative classification models that instead of modelling individual classes (like generative classification models do) try to divide each class from the others. The way they accomplish this is by maximising the distance, or margin, between the division (a line in two-dimensional problems, or a multiplane for n-dimensional ones) and the nearest points, the ones that define the margin and are called support vectors [14]. SVM models can be applied to both classification and regression problems, and have been used in many different areas with great performance especially due to how accurate they are during training and how well they generalise when used on unseen examples [26].

Because SVMs apply a linear separation, it is possible to generalize its use for non-linear applications by projecting the data to a higher dimensional space where the margin can be linearly described. To achieve this, one of many types of kernel transformation functions is used. This use results in a powerful method but, on the other hand, kernel selection and the definition of hyperparameters may be a challenge when building the models [10]. Furthermore, to avoid excessive computational costs associated with the higher dimension calculations, only the scalar products for the expanded dimension are calculated and not the full expansion (this is known as the kernel trick). Of the four most used kernels, in this study we train, test and compare the Linear, Polynomial and Radial Basis Function (RBF) ones.

D. Random Forest Classifiers (RF)

Random Forest classifiers are ensemble learners that use DTs as their basic building blocks. The concept involves overfitting multiple DTs and using a voting or averaging system to achieve a decision; in this way, regions where the classification is less obvious and that could lead to inconsistencies between different classifiers are defined by the decision between multiple trees. The specificity of RF classifiers is that they apply an ensemble of random trees to achieve a better classification and avoid overfitting the data. They do not require scaling or normalization of the data, but to fit the proposed workflow and train an RF after the dimensionality reduction step, we use standardized features in this paper.

E. Principal Component Analysis (PCA)

Principal Component Analysis is an unsupervised learning method mostly used for dimensionality reduction and data visualisation. PCA works with the principle of finding the main axes the data is spread around, and projecting the points into those axes, thus reducing, for example, a 3D dataset to the two principal dimensions, or components, that explain the most variance of data.

The usefulness as a dimensionality reduction tool lies into preserving the components with the maximum variance of data while diminishing the problems associated with the "curse of dimensionality". In other words, applying PCA is useful to reduce the dataset dimensionality while at the same time preserving most of the information and relationship between the points. In this study, running PCA while preserving the high level of 99.9% of the variance has proven

to be a most satisfactory and effective choice to at the same time reduce the dimensionality and preserve information, because in many of the examples modelled, the resulting calculated features are either redundant or not significative.

IV. Experimental Methodology

A. Features Engineering

In most pattern recognition studies, it is assumed that the full event of interest or pattern is included in the observation window, and the same assumption is made here. This study implements two different feature extraction methods, one based only on statistical features using a moving window approach, and for comparison a more complex one that includes statistical, shape, energy, time and frequency domain features, for a total of up to 794 different attributes for each time series.

The moving window concept is very similar to how shape features or moving averages are calculated: each instance is divided into sections, but instead of only using some divisions that split the whole time frame equally, the window of time length l_w slides across the full length in steps of size l_s, and the calculation of the features happens at each step. Reference [11] used a similar approach to break the full sequence into multiple sub-sequences and estimate the moment when the pattern changes but treated each window as a sequence to be identified on its own, and [9] used different sized windows for feature calculation, but not the stepping concept. In this paper, the method is meant to capture the statistical changes of the time series, and the resulting total number of features, given in (1), will be relative to the number of different features n_F, the number of variables n_X, l_w and l_s:

$$n_F \times n_X \times \left(1 + \frac{n - l_w}{l_s}\right) \tag{1}$$

Where in our example the number of different features is seven, the number of variables is one for the SCCP dataset and 17 for the HS data, and n is 60 for both. For this study, we decided to use a very limited number of statistical features, to ensure a low dimensionality and simplify the features engineering process. The statistical features used are the mean and median, standard deviation, skewness, kurtosis, and maximum and minimum values, calculated for every window at each step.

TABLE I. Features extraction setups used in this study

Name	Definition	Name	Definition
X0	Raw data	30-10	t_w = 30, t_s = 10
10-05	t_w = 10, t_s = 5	30-15	t_w = 30, t_s = 15
15-05	t_w = 15, t_s = 5	30-30	t_w = 30, t_s = 30
20-05	t_w = 20, t_s = 5	40-20	t_w = 40, t_s = 20
20-10	t_w = 20, t_s = 10	60-60	the full time-series
20-15	t_w = 20, t_s = 15	TSFRESH1	*extract_features*
		TSFRESH1	*extract_relevant_features*

For comparison with the proposed method, we also extracted two different sets of features using the module *tsfresh* [6]. The first set (that we will call TSFRESH1) was the direct output from the *extract_features* function, and for the SCC data resulted in a total of 779 features. The second set (named TSFRESH2) is the result of an extended functionality called *extract_relevant_features*, that performs an internal features selection step, removing the irrelevant ones based on significance hypothesis tests, and reduced the number above to 396. Table I shows the list of all setups included in this study for the feature extraction phase, including an additional X0 method that used the raw time-series data directly for comparison.

The next step in the data pre-processing was the feature normalisation using the process of standardization or mean normalisation (2) that consists in subtracting from the input point x the mean value (μ) for that variable and dividing the resulting value by its standard deviation (σ). This step is not necessary for Decision Trees or Random Forest Classifiers, but it is highly recommended (and sometimes mandatory) for the linear classification algorithms, as it speeds the convergency and ensures that all features have the same weight.

$$X' = \frac{x - \mu}{\sigma} \tag{2}$$

In the final data preparation step, PCA was used for dimensionality reduction because in general, the performance improves at lower-dimensional space [2]. Instead of plotting the number of principal components versus the explained variance and choosing a number to achieve the desired level of retained variance, as mentioned above, we decided to retain 99.9% of the information. This step proved very effective and essential because, for some of the feature extraction setups, the total number of features was huge as shown in Table II.

TABLE II. Effects of PCA dimensionality reduction for some extraction setups

Feature extraction	dataset	Before PCA	After PCA
t_w = 20, t_s = 10	SCCP	35	26
t_w = 30, t_s = 15	SCCP	21	16
tsfresh1	SCCP	779	264
tsfresh2	SCCP	396	224
t_w = 20, t_s = 10,	HS	595	202
t_w = 30, t_s = 15	HS	357	122
tsfresh1	HS	13243	1997
tsfresh2	HS	6135	1755

B. Testing procedures

We implemented all the machine learning methods described above using the Python programming language and the *scikit-learn* and *tsfresh* modules, running on a personal computer with an Intel Core i7-9850 CPU with 2.60 GHz and 32 GB RAM on Windows 64-bit OS. Before model learning, all datasets were divided into two groups with the same number of samples, one set used for learning and the other for testing. The SCC dataset models used the settings in Table I, while only the best of those have been selected to run the hydraulic systems dataset.

All machine learning methods have several parameters that require calibration. They are called hyperparameters, and the most common tuning method is the grid search [5], that consists of running a full factorial combination between all the parameters chosen for the optimisation [14]. The grid search scoring is usually done with a 10-fold validation approach. This validation technique was proven empirically to be

superior to other procedures [3] and consists of dividing the training set into ten equal parts, the so-called folds, using nine parts for training and the remaining one for validation; then at the next step, the process is repeated with a different part chosen for the validation. The final score is the average score of all ten training plus validation steps and is used for comparison between all combinations of parameter tested [14].

In this paper we decided to use the random search with the 10-fold validation instead, that was proven to be more efficient for hyperparameter optimisation [5], especially in higher dimensional spaces search. With this method, we could run a much finer grid without increasing the number of tests exponentially while saving time by avoiding running many tests in the exploration of dimensions that have less importance.

Besides the individual hyperparameter optimisation for each classifier, we also applied a balanced class weight mode for the LR and SVM algorithms. This choice adjusts each class individual weight with a factor that is inversely proportional to the number of samples in that class and was a decision made to account for the class imbalance present in the HS dataset and that is common in the majority of real-life applications, including control chart pattern recognition ones [22]. This extra design decision should avoid the need for an additional pre-processing step of resampling, that would be either undersampling, by removing data from the majority class, or oversampling, by adding random or artificially generated data to the minority classes. For comparison reasons, we implemented an oversampling procedure known as Synthetic Minority Oversampling Technique (SMOTE) using the *imblearn* module. The reason to avoid this extra step is that the resulting classification problem may either be missing important information (when removing samples) or not generalize well depending on how the extra samples are added, besides increasing the computational cost.

V. Experimental Results

A. Metrics

Accuracy, given by the number of correctly classified examples divided by the total number of samples, may not be an ideal metric on its own, especially when dealing with imbalanced data. Because of this, in this study, we supplement the information obtained by using the accuracy metric with the confusion matrix and other metrics obtained directly from it.

The confusion matrix is a squared matrix of size equal to the number of classes that correlates the classification results in terms of correctly classed examples. For binary classification problems, it looks like the example in Fig.1, where the main diagonal contains the correctly classified examples, labelled TN for true negatives and TP for true positives, and the remaining cells have the incorrectly classified examples, named FN for false negatives and FP for false positives. Based on those values, we can define accuracy as (3):

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3}$$

The other metrics used are precision, recall and the f-score, given by (4) to (6). Precision relates the true positives number to the total of positives (true and false); recall, also called sensitivity, is useful when it is desired to avoid false negatives, as in our examples, because a false negative would mean a failure or process deviation has not been detected. The final metric, f-score, is used to relate precision and recall. Here the variant known as f1 is used, and is the metric presented in this study to complement the accuracy:

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

$$Recall = \frac{TP}{TP + FN} \tag{5}$$

$$F_1 score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{6}$$

B. Results

The best results obtained for the SCC dataset are highlighted in Table III. The moving window method proposed in this paper manages to achieve accuracy results as good as the ones derived from the more elaborated feature extraction methods using *tsfresh*. Both Logistic Regression (LR) and Radial Basis Function Support Vector Machines (RBF SVM) algorithms performed exceptionally well with the 30-15 window, attaining 100% accuracy and correctly classifying all test samples.

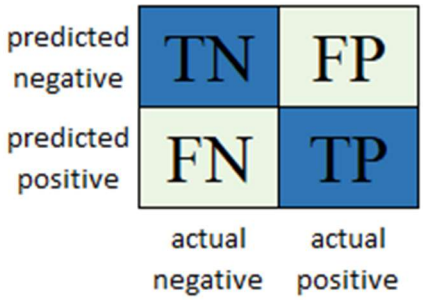

Fig. 1. Example of a confusion matrix for binary classification problems

TABLE III. SCC accuracy results for different feature extraction settings

	SCC							
	X0	***20-05***	***20-10***	***20-15***	***30-10***	***30-15***	***TSFRESH1***	***TSFRESH2***
LR	97.7	99.7	**100.0**	99.7	99.3	**100.0**	**100.0**	**100.0**
DT	86.3	90.0	91.3	89.0	88.7	96.3	97.7	99.0
Linear SVM	78.0	98.3	99.7	98.7	98.0	99.0	99.7	99.7
Poly. SVM	98.7	99.0	98.7	97.3	99.3	98.0	**100.0**	99.7
RBF SVM	98.7	**100.0**	99.7	**100.0**	99.3	**100.0**	**100.0**	**100.0**
RF	98.0	97.7	96.7	96.0	97.3	99.3	98.7	99.0
Average	92.9	97.5	**97.7**	96.8	97.0	**98.8**	**99.4**	**99.6**

The next best scoring window was the 20-10, with an average model accuracy of 97.7%. It did not perform as well as 20-05 or 20-15 with RBF SVM, but also achieved 100% with LR. Considering that the good performance of a machine learning model in a certain dataset does not ensure it will perform well in any other dataset (the so-called "no free lunch" theorem), the average score was used to decide which methods to test on the HS dataset, and the chosen are 20-10, 30-15, TSFRESH1 and TSFRESH2. The confusion matrix for the 20-10 RBF SVM model is shown in Fig. 2. The misclassification happened with examples of downward trend (4) classified as a downward shift (6), what is a common occurrence in the references reviewed.

TABLE IV. ACCURACY / F1 SCORES: WINDOW 20-10 FOR THE HS DATA

	HS_X1-20-10			
	cooler condition		***valve condition***	
	weighted	SMOTE	weighted	SMOTE
LR	99.8 / 1.00	99.7 / 1.00	99.1 / 0.99	98.7 / 0.98
Linear SVM	99.8 / 1.00	99.8 / 1.00	93.2 / 0.92	95.0 / 0.93
Poly. SVM	99.7 / 1.00	**99.9 / 1.00**	97.6 / 0.97	96.0 / 0.95
RBF SVM	99.8 / 1.00	99.8 / 1.00	**99.3 / 0.99**	99.1 / 0.99
	pump leak		***hydraulic accumulator***	
	weighted	SMOTE	weighted	SMOTE
LR	99.0 / 0.99	99.3 / 0.99	91.4 / 0.90	91.0 / 0.90
Linear SVM	94.3 / 0.92	90.8 / 0.87	87.5 / 0.85	84.5 / 0.82
Poly. SVM	96.4 / 0.95	97.0 / 0.96	92.1 / 0.91	92.4 / 0.91
RBF SVM	99.4 / 0.99	**99.7 / 1.00**	96.3 / 0.96	**96.5 / 0.96**

TABLE V. ACCURACY / F1 SCORES: WINDOW 30-15 FOR THE HS DATA

	HS_X1-30-15			
	cooler condition		***valve condition***	
	weighted	SMOTE	weighted	SMOTE
LR	99.9 / 1.00	**100 / 1.00**	97.6 / 0.97	97.3 / 0.97
Linear SVM	99.9 / 1.00	99.8 / 1.00	90.4 / 0.87	91.3 / 0.83
Poly. SVM	99.9 / 1.00	99.9 / 1.00	90.1 / 0.87	89.1 / 0.86
RBF SVM	99.9 / 1.00	99.9 / 1.00	**98.5 / 0.98**	**98.5 / 0.98**
	pump leak		***hydraulic accumulator***	
	weighted	SMOTE	weighted	SMOTE
LR	**99.8 / 1.00**	99.7 / 1.00	88.7 / 0.87	92.5 / 0.91
Linear SVM	96.9 / 0.96	96.2 / 0.95	87.5 / 0.85	89.4 / 0.87
Poly. SVM	96.7 / 0.95	97.4 / 0.96	87.2 / 0.85	88.7 / 0.87
RBF SVM	99.6 / 1.00	99.4 / 0.99	92.6 / 0.91	**94.5 / 0.93**

Many authors compare the results obtained with their proposed methods to those from other references; this comparison with the moving window concept introduced in this study can be found in Appendix B.

To further explore the proposed method, we used the same feature extraction settings that resulted in the best scores when running the SCC data with the HS dataset. Because the Decision Tree and Random Forest classifiers did not perform as well, they are omitted here. The accuracy and F1 scores, including a comparison between weighted SVM and LR and oversampling using SMOTE, are presented in Table IV for window 20-10, Table V for 30-15, and Table VI and Table VII for TSFRESH1 and TSFRESH2 respectively. The bold font values are the best for each condition in each table, and the ones highlighted in blue are the best overall. A general summary is given in Table VIII.

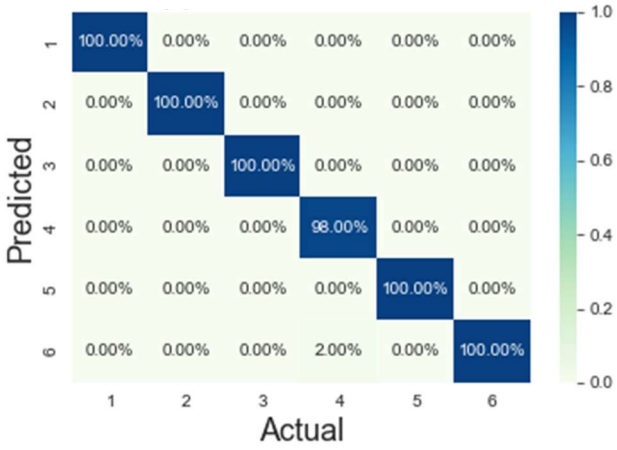

Fig. 2. Confusion matrix for RBF SVM model, 20-10 window, SCC dataset

TABLE VI. ACCURACY / F1 SCORES: TSFRESH1 FOR THE HS DATA

	HS_X2_TSFRESH1			
	cooler condition		***valve condition***	
	weighted	SMOTE	weighted	SMOTE
LR	**100 / 1.00**	99.8 / 1.00	80.2 / 0.78	81.3 / 0.79
Linear SVM	**100 / 1.00**	99.7 / 1.00	72.3 / 0.69	68.4 / 0.65
Poly. SVM	99.9 / 1.00	99.9 / 1.00	70.0 / 0.62	72.5 / 0.67
RBF SVM	99.5 / 1.00	99.5 / 0.99	**83.2 / 0.81**	82.2 / 0.79
	pump leak		***hydraulic accumulator***	
	weighted	SMOTE	weighted	SMOTE
LR	93.3 / 0.90	91.5 / 0.88	90.5 / 0.89	90.3 / 0.88
Linear SVM	89.8 / 0.86	88.0 / 0.85	82.6 / 0.79	82.9 / 0.80
Poly. SVM	80.6 / 0.74	82.1 / 0.75	86.9 / 0.85	89.2 / 0.88
RBF SVM	**93.7 / 0.91**	93.1 / 0.90	95.1 / 0.94	**95.8 / 0.95**

Once again, running with the HS data this time, the moving window method obtains results as good as or even better than the ones using *tsfresh* features extraction, the only exception being the hydraulic accumulator condition. In this case, using TSFRESH2 weighted resulted in 96.9 % accuracy with RBF SVM while the best-weighted window is the 20-10, also with RBF SVM, at 96.3%. For the cooler condition, TSFRESH1 weighted achieved 100% with LR and Linear SVM (the same value as the 30-15 window with SMOTE and LR), but all 30-15 weighted classifiers resulted in 99.9%, indicating better robustness for the moving window method. Finally, both the valve condition and pump leak models obtained higher results using the proposed approach: 99.3% (20-10 weighted RBF SVM) versus 86.9% (TSFRESH2 SMOTE RBF SVM) for the former, and 99.8% (30-15 weighted LR) versus 98.6 (TSFRESH2 SMOTE RBF SVM) for the latter.

Considering the strategy to deal with unbalanced datasets, the weighted choice attains better results overall than using the

SMOTE oversampling method. Only in a few tests, the results obtained with oversampled data are better than the ones using the weight factor proportional to the number of examples in each class; and when this happens, the only example where SMOTE achieved the global best was for the cooler condition, 30-15 window with Linear Regression.

TABLE VII. ACCURACY / F1 SCORES: TSFRESH2 FOR THE HS DATA

	HS_X2_TSFRESH2			
	cooler condition		***valve condition***	
	weighted	SMOTE	weighted	SMOTE
LR	**99.9 / 1.00**	99.8 / 1.00	83.9 / 0.83	81.7 / 0.81
Linear SVM	**99.9 / 1.00**	**99.9 / 1.00**	79.8 / 0.78	79.2 / 0.76
Poly. SVM	99.8 / 1.00	99.7 / 1.00	84.8 / 0.83	81.1 / 0.79
RBF SVM	**99.9 / 1.00**	**99.9 / 1.00**	84.3 / 0.83	**86.9 / 0.85**
	pump leak		***hydraulic accumulator***	
	weighted	SMOTE	weighted	SMOTE
LR	96.6 / 0.95	96.2 / 0.95	92.8 / 0.91	92.6 / 0.91
Linear SVM	93.6 / 0.91	93.7 / 0.91	86.9 / 0.85	84.2 / 0.82
Poly. SVM	93.4 / 0.91	93.7 / 0.91	94.7 / 0.94	94.1 / 0.93
RBF SVM	98.3 / 0.97	**98.6 / 0.98**	**96.9 / 0.96**	96.2 / 0.96

TABLE VIII. SUMMARY OF BEST RESULTS FOR HS DATA

HS				
target condition	algorithm	strategy	accuracy	F_1 score
cooler condition	LR, linear SVM	TSFRESH1 weighted, 30-15 SMOTE	100.0	1.00
valve condition	RBF SVM	20-10 weighted	99.3	0.99
pump leak	LR	30-15 weighted	99.8	1.00
hydraulic accumulator	RBF SVM	TSFRESH2 weighted	96.9	0.96

When taking into consideration only the results from the proposed method, other than the pump leakage using 30-15 window, all the best scores were obtained with RBF-SVM. Fig. 3 shows the confusion matrixes for the 20-10 RBF-SVM tests, and Fig. 4 for the 30-15 window, and a comparison with results from the reference can be found in Appendix B.

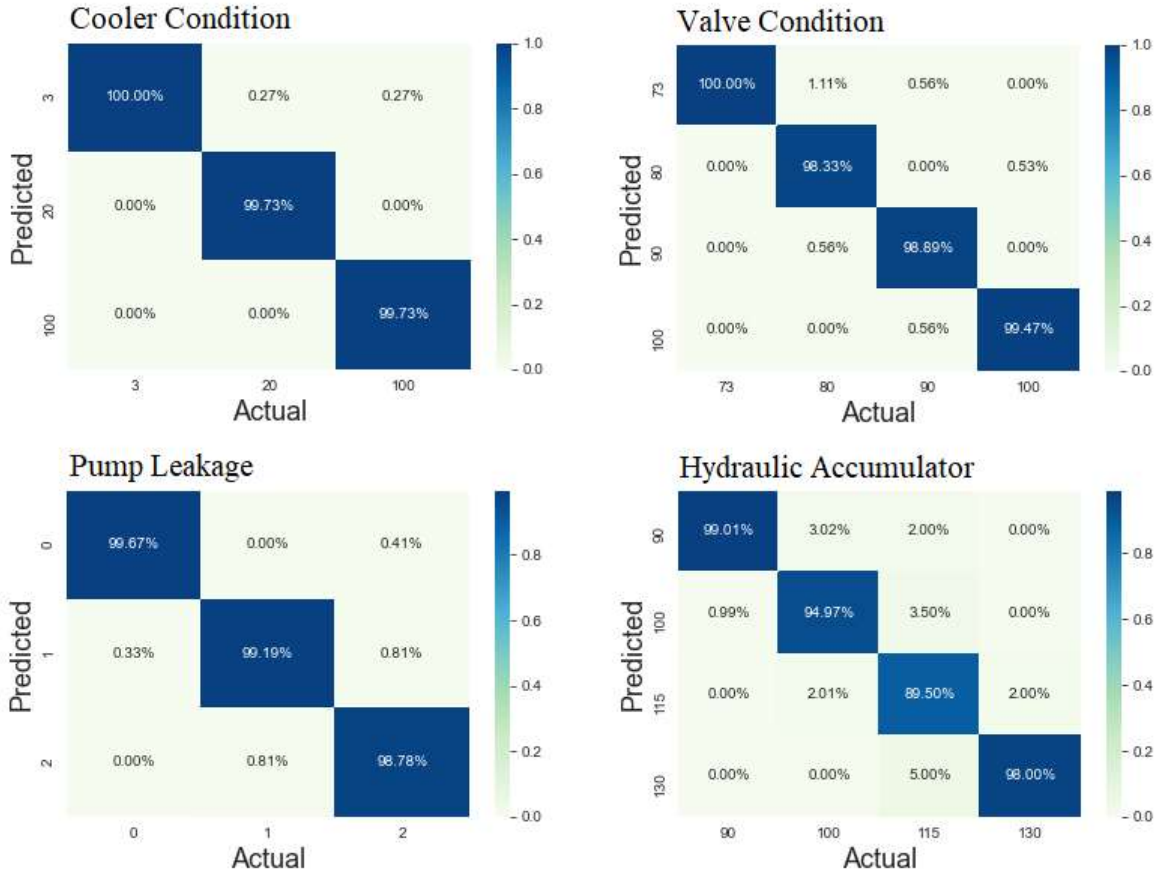

Fig. 3. 20-10 RBF SVM confusion matrix for the HS dataset

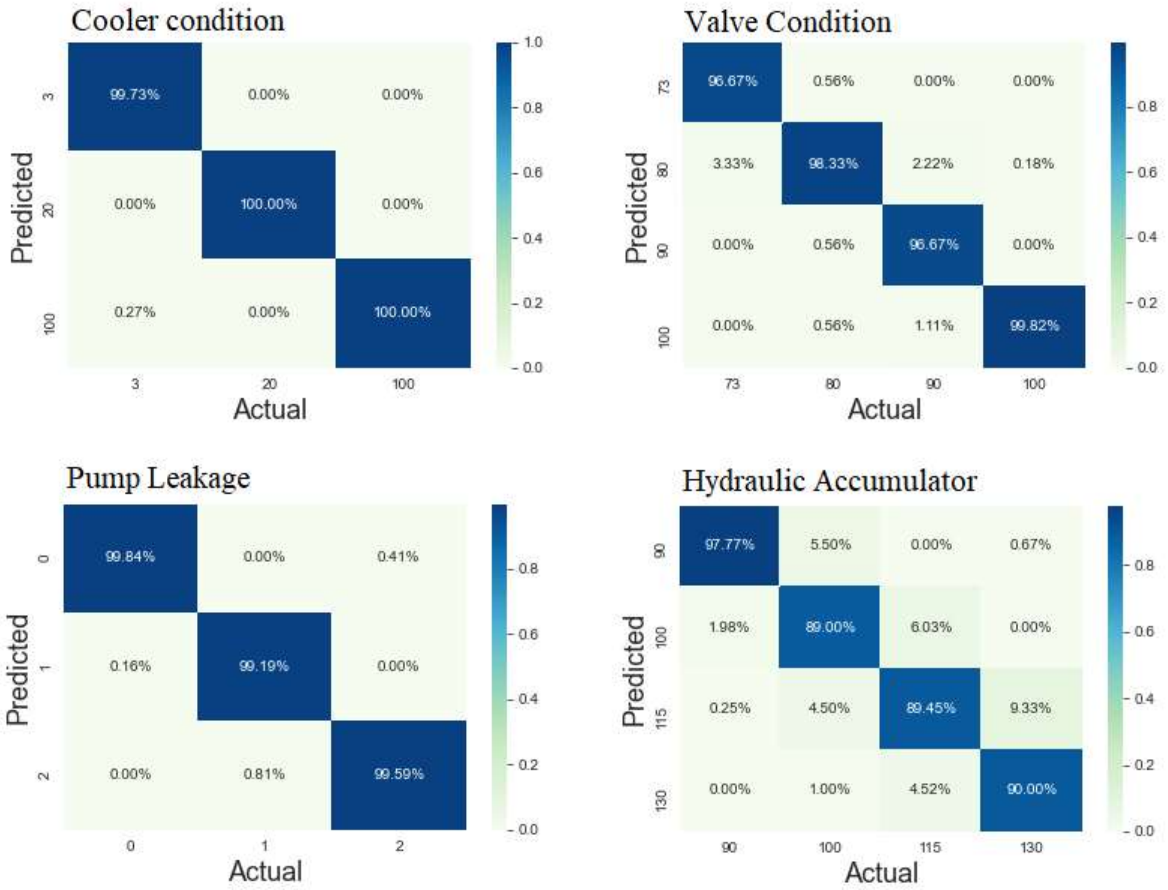

Fig. 4. 30-15 RBF SVM confusion matrix for the HS dataset

VI. Final Discussion

RBF-SVM is, overall, the most successful algorithm in this implementation using the moving window approach. The examples when the best score does not result from it, the value is usually a close second. This probably happens because the classes are naturally either normally distributed or very close to it and the spherical and ellipsoidal decision areas produced by this kernel improves the fit [22].

In this paper, we used the same approach and exactly same modelling, learning and testing steps in two very different datasets: one from the simulation of control chart patterns and another from readings of actual sensors obtained from a hydraulic system test rig. In both cases, the proposed method, using balanced weighting factors with the moving windows feature engineering approach, has shown excellent results when compared with the reviewed literature and the features extracted using the module *tsfresh*. It can be easily adapted for use in real-time Industry 4.0 applications, monitoring production process as well as a range of sensors from machinery and IoT devices. The window, step and total lengths can be easily adapted to a broad range of implementations, as well as a more comprehensive number of features can be calculated to better capture the specific characteristics of every application.

Another advantage of the proposed approach is that it is less computationally expensive once the features and windows are choosen: as shown before, the *tsfresh* extraction results in many more features than are needed, and the process itself takes much longer than the moving window calculations. For the cases when the resulting score is not satisfactory, more features could have been calculated and by using the forward feature selection method, or other similar, features that do not contribute significantly would be removed.

The definition of the features to be calculated requires some expert knowledge, what could be a disadvantage of using selected features instead of the raw data according to many of the references mentioned in the introduction, most notably the ones that used some form or another of Neural Networks to define, automatically, the features. But it could also be beneficial in creating more efficient and precise models, as technical knowledge of the system can help narrow down the design options for a machine learning implementation.

As suggestions for future work, the framework proposed here could be implemented in a prognostic and predictive maintenance problem or adapted for streaming or real-time data classification. After training, the whole window length could be moved in a second-by-second frame, and as soon as the condition changes, should be able to detect those changes; many publicly available datasets or the SCC equations could be used for development and testing purposes. Another possibility would be detecting the transient phase before an actual failure occurs: if labelled data is available, the model could be trained using purely with no failure and purely with failure data and would be able to classify transient data that is close enough to the failure data, therefore detecting the errors before they occur, using a semi-supervised approach.

REFERENCES

[1] Abouel Nasr, E. S., & Al-Mubaid, H. (2009). (2009). Mining process control data using machine learning. Paper presented at the - 2009 International Conference on Computers & Industrial Engineering, 1434-1439. https://doi.org/10.1109/ICCIE.2009.5223783

[2] Addeh, A., Khormali, A., & Golilarz, N. A. (2018). Control chart pattern recognition using RBF neural network with new training algorithm and practical features. ISA Transactions, 79, 202-216. https://doi.org/https://doi.org/10.1016/j.isatra.2018.04.020

[3] Ahmed, N. K., Atiya, A. F., Gayar, N. E., & El-Shishiny, H. (2010). An empirical comparison of machine learning models for time series forecasting.29(5-6), 594-621. https://doi.org/10.1080/07474938.2010.481556

[4] Alcock, R. J., & Manolopoulos, Y. (1999). Time-series similarity queries employing a feature-based approach. 7th Hellenic Conference on Informatics,

[5] Bergstra, J., & Bengio, Y. (2012). Random search for hyper-parameter optimization. J.Mach.Learn.Res., 13, 281–305.

[6] Christ, M., Braun, N., Neuffer, J., & Kempa-Liehr, A. W. (2018). Time series feature extraction on basis of scalable hypothesis tests (tsfresh – A python package). Neurocomputing, 307, 72-77. https://doi.org/https://doi.org/10.1016/j.neucom.2018.03.067

[7] Gauri, S. K., & Chakraborty, S. (2009). Recognition of control chart patterns using improved selection of features. Computers & Industrial Engineering, 56(4), 1577-1588. https://doi.org/https://doi.org/10.1016/j.cie.2008.10.006

[8] Golilarz, N. A., Addeh, A., Gao, H., Ali, L., Roshandeh, A. M., Munir, H. M., & Khan, R. U. (2019). A new automatic method for control chart patterns recognition based on ConvNet and Harris hawks meta heuristic optimization algorithm https://doi.org/10.1109/ACCESS.2019.2945596

[9] Helwig, N., Pignanelli, E., & Schütze, A. Condition monitoring of a complex hydraulic system using multivariate statistics. Paper presented at the - 2015 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings, 210-215. https://doi.org/10.1109/I2MTC.2015.7151267

[10] Hong, Z., Li, Y., & Zeng, Z. (2019). (2019). Convolutional neural network for control chart patterns recognition. Paper presented at the Proceedings of the 3rd International Conference on Computer Science and Application Engineering, Sanya, China. https://doi.org/10.1145/3331453.3360974

[11] Hu, S., L. Zhao, & Zhao, L. (2015). (2015). A support vector machine based multi-kernel method for change point estimation on control chart. Paper presented at the - 2015 IEEE International Conference on Systems, Man, and Cybernetics, 492-496. https://doi.org/10.1109/SMC.2015.97

[12] imbalanced-learn. (n.d.). Welcome to imbalanced-learn documentation! imbalanced-learn v0.7.0 documentation. Retrieved November 20, 2020, from https://imbalanced-learn.org/

[13] McKinney, W. (2017). Python for data analysis: Data wrangling with pandas, NumPy, and IPython (Second edition. ed.)

[14] Müller, A. C., & Guido, S. (2017). Introduction to machine learning with python: A guide for data scientists. O'Reilly Media.

[15] Pham, D. T., & Chan, A. B. (1998). Control chart pattern recognition using a new type of self-organizing neural network. Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering, 212(2), 115-127. https://doi.org/10.1243/0959651981539343

[16] Ranaee, V., Ebrahimzadeh, A., & Ghaderi, R. (2010). Application of the PSO-SVM model for recognition of control chart patterns. ISA Transactions, 49(4), 577—586. https://doi.org/10.1016/j.isatra.2010.06.005

[17] Scikit-learn. (n.d.). Machine Learning in Python — scikit-learn 0.23.2 documentation. Retrieved November 20, 2020, from https://scikit-learn.org/stable/

[18] Sktime. (n.d.). sktime v0.4.3 documentation. Retrieved November 20, 2020, from https://www.sktime.org/

[19] Tsfresh. (n.d.). tsfresh — tsfresh v0.17.1. documentation. Retrieved November 20, 2020, from https://tsfresh.readthedocs.io/

[20] UCI machine learning repository: Condition monitoring of hydraulic systems data set. https://archive.ics.uci.edu/ml/datasets/Condition+monitoring+of+hydraulic+systems

[21] UCI machine learning repository: Synthetic control chart time series data set. http://archive.ics.uci.edu/ml/datasets/Synthetic+Control+Chart+Time+Series

[22] Xanthopoulos, P., & Razzaghi, T. (2014). A weighted support vector machine method for control chart pattern recognition. Computers & Industrial Engineering, 70, 134-149. https://doi.org/https://doi.org/10.1016/j.cie.2014.01.014

[23] Xu, J., Lv, H., Zhuang, Z., Lu, Z., Zou, D., & Qin, W. (2019). Control chart pattern recognition method based on improved one-dimensional convolutional neural network. IFAC-PapersOnLine, 52(13), 1537-1542. https://doi.org/https://doi.org/10.1016/j.ifacol.2019.11.418

[24] Zan, T., Liu, Z., Su, Z., Wang, M., Gao, X., & Chen, D. (2019). Statistical process control with intelligence based on the deep learning model. Applied Sciences, 10, 308.

[25] Zan, T., Liu, Z., Wang, H., Wang, M., & Gao, X. (2020). Control chart pattern recognition using the convolutional neural network. Journal of Intelligent Manufacturing, 31https://doi.org/10.1007/s10845-019-01473-0

[26] Zhao, C., Wang, C., Hua, L., Liu, X., Zhang, Y., & Hu, H. (2017). Recognition of control chart pattern using improved supervised locally linear embedding and support vector machine. Procedia Engineering, 174, 281-288. https://doi.org/https://doi.org/10.1016/j.proeng.2017.01.138

APPENDICES

All Python 3 code produced, and the results of every test run are stored and fully available in my GitHub page: https://github.com/LFL077/7072CEM.

A. Datasets Information

In Table IX, μ and σ represent, respectively, the process mean and standard deviation when it is under control, r is the random noise at the sampling time, T is the period and a the amplitude of the cycle, g is the slope of the increasing or decreasing trends, and x is the shift magnitude.

TABLE IX. SCC EQUATIONS AND PARAMETERS [4] AND [15].

Class	Description	Equation	Values
1	Normal (NOR)	$y(t) = \mu + r \times \sigma$	$\mu = 30, \sigma = 2,$ $-3 \le r \le 3$ $10 \le a \le 15$ $10 \le T \le 15$ $0.2 \le g \le 0.5$ $7.5 \le x \le 20$ $\frac{n}{3} \le t_3 \le \frac{2n}{3}$ $n = 60\ samples$
2	Cyclic (CYC)	$y(t) = \mu + r \times \sigma + a \times \sin\left(\frac{2\pi t}{T}\right)$	
3	Upward Trend (UT)	$y(t) = \mu + r \times \sigma + g \times t$	
4	Downward Trend (DT)	$y(t) = \mu + r \times \sigma - g \times t$	
5	Upward Shift (US)	$y(t) = \mu + r \times \sigma + k \times x$ $k = 1\ if\ t \ge t_3, else\ k = 0$	
6	Downward Shift (DS)	$y(t) = \mu + r \times \sigma - k \times x$ $k = 1\ if\ t \ge t_3, else\ k = 0$	

TABLE X. ATTRIBUTE INFORMATION OF THE HS DATASET [9].

Sensor	Physical quantity	Unit	Sampling rate
PS1	Pressure	bar	100 Hz
PS2	Pressure	bar	100 Hz
PS3	Pressure	bar	100 Hz
PS4	Pressure	bar	100 Hz
PS5	Pressure	bar	100 Hz
PS6	Pressure	bar	100 Hz
EPS1	Motor power	W	100 Hz
FS1	Volume flow	l/min	10 Hz
FS2	Volume flow	l/min	10 Hz
TS1	Temperature	°C	1 Hz
TS2	Temperature	°C	1 Hz
TS3	Temperature	°C	1 Hz
TS4	Temperature	°C	1 Hz
VS1	Vibration	mm/s	1 Hz
CE	Cooling efficiency (virtual)	%	1 Hz
CP	Cooling power (virtual)	kW	1 Hz
SE	Efficiency factor	%	1 Hz

Table X lists all sensors and their respective unit and sampling rate (frequency) used to generate the HS dataset, while Table XI lists the target conditions for the same dataset next to their values (and the unit, if applicable).

TABLE XI. TARGET CONDITIONS OF THE HS DATASET (VALUES AND NUMBER OF INSTANCES) [9].

1: Cooler condition / %:		**# of instances**
3	close to total failure	732
20	reduced efficiency	732
100	full efficiency	741
2: Valve condition / %:		
100	optimal switching behaviour	1125
90	small lag	360
80	severe lag	360
73	close to total failure	360
3: Internal pump leakage:		
0	no leakage	1221
1	weak leakage	492
2	severe leakage	492
4: Hydraulic accumulator / bar:		
130	optimal pressure	599
115	slightly reduced pressure	399
100	severely reduced pressure	399
90	close to total failure	808

B. Comparison with selected results from other references

Table XII shows a comparison between the proposed method and some selected values found in other references for the control chart pattern recognition problem. The accuracy score for the moving window approach is the average between the five best windows (20-05, 20-10, 20-15, 30-10, 30-15) for the classifier with the highest score. This value would be 100% when considering only the best overall results.

TABLE XII. SCC DATASET ACCURACY SCORE COMPARISON

Reference	Method	# of Patterns	Total Accuracy
[16]	Selected features and PSO-SVM	9	99.58
[26]	Selected Features and SLLE-SVM	7	99.43
[2]	Selected features and Bees-RBF	8	99.63
[8]	Raw data and HHO-ConvNet	6	99.80
[23]	Raw data and 1D-CNN	8	98.96
[24]	Raw data and 1D-CNN	9	99.26
[26]	Raw data and 1D-CNN	6	99.30
This study	Moving window and RBF-SVM	6	99.80
This study	*tsfresh* and RBF-SVM or LR	6	100.00

For the hydraulic system dataset, Table XIII shows a comparison with the original paper [9]. It is only fair to mention that in the original reference, the authors also increased the number of features and run with many different time windows, achieving an improvement of about 25% for the pump leakage and 40% for the hydraulic accumulator pressure loss classifications when using the LDA method.

TABLE XIII. CONDITION MONITORING OF HYDRAULIC SYSTEMS DATASET ACCURACY SCORES COMPARISON

	LDA from reference	**RBF SVM from reference**	**20-10 (RBF SVM weighted)**	**30-15 (RBF SVM weighted)**
Cooler	100	100	99.8	99.9
Valve	100	95.7	99.3	98.5
Pump	73.6	64.2	99.4	99.6
Accumulator	54.0	65.7	96.3	92.6
Mean	81.9	81.4	98.7	97.7