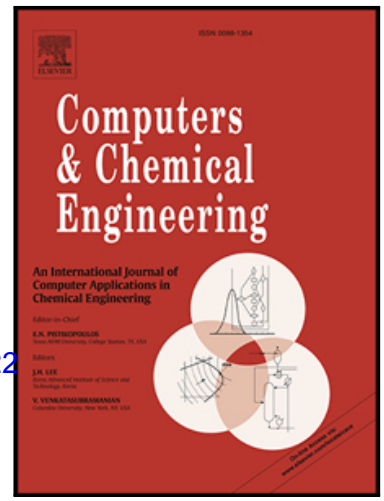

Predicting Chattering Alarms: a Machine Learning Approach

Nicola Tamascelli , Nicola Paltrinieri , Valerio Cozzani

PII: S0098-1354(20)30800-0
DOI: https://doi.org/10.1016/j.compchemeng.2020.107122
Reference: CACE 107122

To appear in: *Computers and Chemical Engineering*

Received date: 29 July 2020
Revised date: 30 September 2020
Accepted date: 5 October 2020

Please cite this article as: Nicola Tamascelli , Nicola Paltrinieri , Valerio Cozzani , Predicting Chattering Alarms: a Machine Learning Approach, *Computers and Chemical Engineering* (2020), doi: https://doi.org/10.1016/j.compchemeng.2020.107122

Predicting Chattering Alarms: a Machine Learning Approach

Nicola Tamascelli[1,2], Nicola Paltrinieri[2,*], Valerio Cozzani[1]

[1] Department of Civil, Chemical, Environmental and Materials Engineering, University of Bologna, Bologna, Italy

[2] Department of Mechanical and Industrial Engineering, NTNU, Trondheim, Norway

* nicola.paltrinieri@ntnu.no

Abstract

Alarm floods represent a widespread issue for modern chemical plants. During these conditions, the number of alarms may be unmanageable, and the operator may miss safety-critical alarms. Chattering alarms, which repeatedly change between the active and non-active states, are responsible for most of the alarm records within a flood episode. Typically, chattering alarms are only addressed and removed retrospectively (e.g. during periodic performance assessments). This study proposes a Machine-Learning based approach for alarm chattering prediction. Specifically, a method for dynamic chattering quantification has been developed, whose results have been used to train three different Machine Learning models – Linear, Deep, and Wide&Deep models. The algorithms have been employed to predict future chattering behavior based on actual plant conditions. Performance metrics have been calculated to assess the correctness of predictions and to compare the performance of the three models.

Keywords: *Machine Learning, Data Mining, Alarm management, Alarm floods, Chattering alarms, Chattering prediction*

Definitions

Accuracy	the ratio of number of correct predictions to total number of predictions.
Alarm flood	a condition during which the alarm rate is greater than the operator can effectively manage (e.g., more than 10 alarms per 10 minutes).
Alarm identifier	a code defining the alarm status.
Alarm source	a field device, control system, or Human Machine Interface that can trigger a change in the alarm status.
Binary Database	a database where unique alarms data are represented as sequences of 0's and 1's at one-second sampling.
Chattering alarm	an alarm that repeatedly transitions between the active and the not active states in a short period (e.g., 3 or more alarm records in one minute).

Chattering Index	an index to quantify the amount of chattering that a unique alarm has shown over a certain time period.
Dynamic Chattering Index	an index to quantify the amount of chattering that a unique alarm has shown up to one hour after each alarm event.
Example	the description of an alarm event in terms of features and related label.
Feature	a meaningful attribute of an alarm event.
Label	the category of an alarm event – “Y” for “Chattering within one hour”, “N” for “Not Chattering within one hour”.
Machine Learning	computational methods using experience to improve performance or to make accurate predictions.
Nuisance alarm	an alarm that annunciates excessively, unnecessarily, or does not return to normal after the operator action is taken.
Precision	the fraction of positively predicted labels that are, in fact, positive.
Probability Threshold	an adjustable parameter used to convert raw predicted probabilities into predicted labels.
Recall	the fraction of real positive labels correctly predicted.
Run Length	the time difference in seconds between two consecutive alarm events from the same unique alarm.
Unique alarm	the unique combination of an alarm *source* and an *identifier*.
Unlabeled examples	the description of an alarm event in terms of a list of features.

1. Introduction

The digital revolution and the advent of Distributed Control Systems have undeniably improved the flexibility of industrial alarm systems (Shaw, 1993). Installing new alarms has become relatively simple and economical (Katzel, 2007), but the misconception that more alarms would improve safety and reliability persists in some cases. On the contrary, too many alarms can negatively affect the performance of the alarm system and prevent an adequate operator's response (Kondaveeti et al., 2013; Laberge et al., 2014). Unsatisfactory alarm rationalization is expressed by episodes where an excessive number of alarms are triggered in a short period (ANSI/ISA, 2016; EEMUA, 2013; Laberge et al., 2014). A specific term is coined to define a period of intense alarm activity – an "alarm flood" (Beebe et al., 2012). Hundreds or even thousands of alarms may be triggered during a flood episode, causing a substantial distraction to the operators, and increasing the risk of missing critical alarms (Laberge et al., 2014).

Several studies, accident reports, and standard manuals have cited alarm floods as a contributing factor to financial loss, injuries, and deaths in the chemical industry (Beebe et al., 2012; EEMUA, 2013; Stanton and Barber, 1995), including the investigation report on the explosion in Pembroke Refinery on the 24 July 1994 (Health and Safety Executive, 1997). The accident was caused by a faulty control valve, which was stuck in a closed position. Unfortunately, the control system erroneously indicated that the valve was open, and operators had not been able to identify the problem. Due to the blocked valve, the liquid had accumulated inside a debutanizer column, causing the pressure to increase over the PSV setpoint. A liquid-vapor stream entered the flare pipe that eventually broke since it was not designed to handle liquids. Roughly 10 to 20 tons of partially vaporized flammable materials were released and mixed with air, forming a flammable cloud that ignited and exploded 4 hours after the valve failure, and 20 seconds after the pipe rupture. As a consequence, 26 workers were injured, and the refinery was severely damaged: £48 million were spent on rebuilding the damaged plant, to which the costs of prolonged business interruption should be added. During the accident, alarms were notified to the operators at the rate of one every two to three seconds. Approximately 275 alarms were triggered in the last eleven minutes before the accident, without a concrete effect on the possibility of preventing the accident.

Most of the alarm events within a flood episode are produced by alarms that oscillate between the active and not active state with high frequency –i.e., chattering alarms. Standard manuals have been published (ANSI/ISA, 2016; EEMUA, 2013), providing guidelines for proper alarm rationalization and management, suggesting strategies for chattering and floods reduction. Still, chattering alarms are only addressed and removed retrospectively. Rather than addressing the problem after chattering has happened, a method to predict future chattering based on actual process conditions would significantly improve the performance of the alarm system. Nevertheless, predictive methods based on first principles would be complicated to obtain because many variables influence the dynamics of the system (Ahmed et al., 2013). In this multivariate context, a statistical data-based approach appears to be more feasible. Chemical plants

produce a large quantity of process and alarm data on a daily basis (Reis and Kenett, 2018). Thus, the use of Machine Learning techniques appears to be an interesting opportunity to extract knowledge from these data and to build predictive models. Various researches have focused on the development of Machine Learning algorithms for fault detection and diagnosis (Mahadevan and Shah, 2009; Miao et al., 2013; Zhong et al., 2014), risk assessment (Paltrinieri et al., 2019), process simulation (Aleixandre et al., 2015; Zhang et al., 2010), and dimensionality reduction (Ge et al., 2017). However, to the best of our knowledge, there is not a direct application of these algorithms for alarm chatter prediction.

The present study proposes a Machine Learning approach for chattering prediction. An industrial alarm database has been used to support the analysis. Initially, a modified version of the Chattering Index proposed by Kondaveeti *et al.* (2013) has been developed and used to classify historical alarm events as "Chattering within an hour" or "Not Chattering within an hour" (i.e., alarms that will/will not show chattering within one hour after an alarm event). The results of this method, named Dynamic Chattering Index, have been used to train and evaluate three different Machine Learning classification models –i.e., Linear, Deep, and Wide&Deep models. Each algorithm has been trained and assessed independently on the same dataset. Performance metrics have been calculated to assess the correctness of predictions and to compare the performance of the three models.

The paper is organized in 8 Sections. Section 2 provides an overview of industrial alarms and alarm databases, including definitions of nuisance alarms, chattering, and alarm floods. Section 3 focuses on the database used in this work; a brief description of the plant section that generated the alarms is also provided. Section 4 describes the methodology, which includes the preprocessing of alarm data, the development of the Dynamic Chattering Index, the Machine Learning models, and the performance metrics that have been used to evaluate the models. Section 5 provides a detailed description of the Machine Learning simulations. The results of the simulations are presented in Section 6 and discussed in Section 7. Finally, conclusions are summarized in Section 8.

2. Alarms in the chemical industry

Disturbances of various nature cause inherent process fluctuation during daily operations. Typically, minor deviations are managed by the Basic Process Control System (BPCS), and process oscillations are maintained to an acceptable level. However, situations may arise where automatic systems fail to restore normal operations, and human intervention is needed. In these circumstances, alarms inform the operator that process conditions are significantly deviating from their normal operating state (ANSI/ISA, 2016). Each alarm should support a timely and effective response by providing guidance to a set of corrective actions.

2.1. Nuisance, chattering and alarm floods

If an alarm does not convey any new information, or if no corrective action is possible, the alarm is ineffective. These types of alarms are called “nuisance” and are often caused by poorly managed alarm systems (ANSI/ISA, 2016; Kondaveeti et al., 2010). Different types of nuisance alarms can be identified (e.g., Chattering, Fleeting, Stale alarms)(ANSI/ISA, 2016), but in this study the attention has been directed to chattering alarms – i.e., alarms that “repeatedly transitions between the active state and the not active state in a short period of time” (ANSI/ISA, 2016). A rule of thumb to determine chattering behavior is three or more alarm records (from the same alarm source) in one minute (Kondaveeti et al., 2010).

Besides, alarm floods –i.e., periods when the alarm rate exceeds 10 alarms/operator per ten minutes time interval– are another common issue in modern alarm systems (ANSI/ISA, 2016; Laberge et al., 2014). Due to the intense alarm activity, hundreds of alarm records may be produced in a short time. The workload caused by flood episodes is often overwhelming: the operator cannot provide an appropriate response, and crucial alarms are likely to be missed (Ahmed et al., 2013). Usually, most of the alarms inside a flood episode come from a limited number of alarm sources (ANSI/ISA, 2016). Furthermore, alarm floods are strongly related to chattering alarms due to their potential to cause a large number of alarm events in a short time span. For this reason, identifying and removing chattering alarms is a crucial step to improve the performance of the alarm system and to avoid flood episodes.

2.2. Alarm attributes

Alarm events are described through a list of attributes. Each attribute defines a characteristic of an event such as the time of the alarm occurrence (i.e., the *Timestamp*), the *Source* that triggered the alarm, the alarm status, and more. Table 1 describes a list of attributes that are most frequently presented to the operator. It is worth mentioning that different companies use different messages and different sets of alarm attributes. The table is thus a selection of the most common and significant alarm attributes.

Attribute	Description
Timestamp	Date and time (GMT) of the alarm event.
Source	The source that triggered the alarm. It might be a measuring instrument or a PLC function.
Jxxx	The safety interlock logic associated with the alarm, where “xxx” is a three digits code.
Alarm Identifier	A code that defines the alarm status (e.g. “HHH”, “HTRP”, “LLL”, “IOP”, “HHH Recover”, “ACK”).
Data Value	The value of the process variable.
Eng. Unit	The units of measure of the process variable (e.g. “ % ”, “ °C ”, “ KPa ”).

Table 1 - Selection of the most common and significant alarm attributes presented to the operator.

When an alarm is triggered, a message appears on the operator console. An example is:

"LI01 LEVEL D01 J434 PV = 98,0 % HHH"

The alarm message reports the source of the alarm (the level indicator 01), a brief explanation of the measured variable (the level in drum 01), the associated safety function (J434), the value of the process value (98 %) and finally, the alarm status (High Level –i.e., "HHH").

For a more comprehensive understanding of the following analyses, the *alarm identifier* must be described more in detail. The identifiers "HHH" and "HTRP" inform that the measured variable has exceeded the "high" and "very high" threshold respectively, "LLL" and "LTRP" refer to the "low" and "very low" threshold, "IOP" informs about an instrumental failure or out-of-range measure, "ACK" indicates that the operator has acknowledged the alarm. The alarm identifier may include the word "Recover" (e.g. "HHH Recover", "LTRP Recover"), that indicates that the original alarm has been recovered (i.e., the alarm is not active anymore). In addition, two more attributes must be described, the Active Time Delta and the Condition Name, which have been used in the analyses but are not listed in Table 1. The Active Time Delta (ATD) is the number of seconds between an alarm and its recovery. The Condition Name (CN) is the alarm identifier of the initial alarm event (e.g., if the alarm is an "LLL Recover", CN will be "LLL").

In spite of the variety of different attributes, an alarm event is uniquely identified by three attributes only (Kondaveeti et al., 2010):

1. Time Stamp;
2. Source;
3. Alarm Identifier.

Also, the combination of an alarm *source* and an *identifier* (e.g., "LI01 HHH", "PI103 LTRP") is called a *unique alarm* (Kondaveeti et al., 2010).

2.3. Alarm databases

Chemical plants produce a massive amount of data on a daily basis (Kordic et al., 2010). Alarm events are continuously recorded and stored in alarm databases, which are characterized by a large search-space and may contain years of alarm data (Kordic et al., 2010). Typically, alarm events are collected as chronologically ordered time sequences (Weiss, 2010). Each row of the database represents an event, and each column represents an attribute of the alarm event. Obviously, there is not a single database format: different companies use different Distributed Control Systems (DCS). The format, the codes and the set of displayed features may vary accordingly. Typically, an alarm database contains more features than those presented in Table 1, but most of these additional features are either redundant or not useful for the analyses.

The analysis of the alarm history is a crucial step in monitoring the alarm system performance (ANSI/ISA, 2016). Periodic study of the alarm database allows the production of performance metrics and the detection of nuisance alarms. An example of a performance metric suggested by ANSI/ISA (2016) is the "Percentage of time the alarm system is in a flood condition", which must be lower than 1 % to grant stable operations. Furthermore, the standard states that chattering alarms must not be tolerated, and actions must be taken to resolve any chattering that occurs. Nevertheless, due to the complexity and quantity of data in alarm databases, the extraction of relevant information is not trivial and usually requires time and resources (Kordic et al., 2010).

3. Case-study: ammonia production plant layout and alarms

The industrial alarm database used in this study is provided by an international chemical company and consists of alarm data that were collected in a plant section for ammonia synthesis. The process involves the manipulation of a significant amount of dangerous substances (e.g., methane, hydrogen, ammonia), and severe operating conditions are often required (e.g., high temperature, high pressure, corrosive fluids). According to the European Parliament and Council Directive 2012/18/EU (European Union, 2012), the plant has been classified as an upper-tier establishment due to its potential to cause major accidents.

The ammonia production plant comprises four sections:

1. Desulfurization and Reforming;
2. Water-Gas Shift, CO_2 Removal, and Methanation;
3. Ammonia synthesis and Cooling circuit;
4. Anhydrous ammonia storage, Pipeline, and Loading/unloading tankers.

Figure 1 shows a schematic representation of the plant layout for ammonia production, excluding storage, loading, and unloading (section 4).

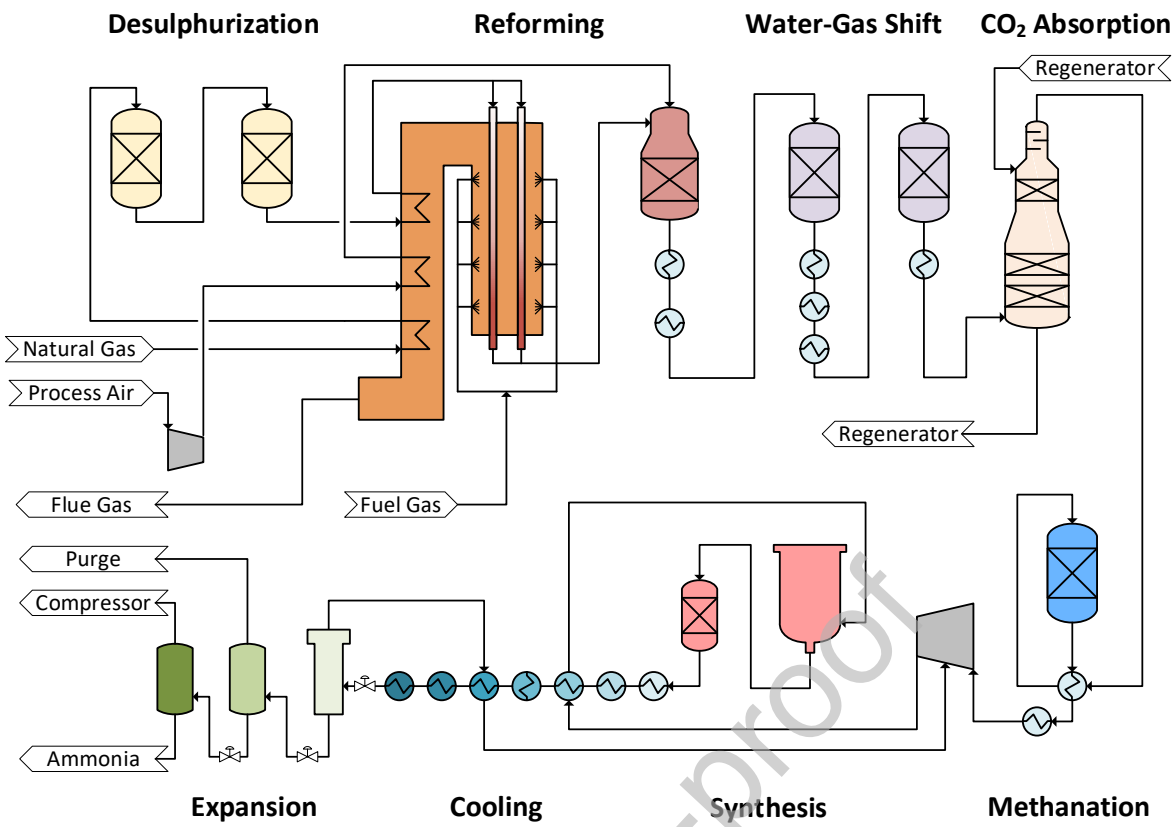

Figure 1 – Simplified process scheme of the ammonia production plant considered.

Natural Gas, Air, and Steam are used as raw materials for ammonia synthesis, according to the following exothermic reaction:

$$N_2 + 3\,H_2 \rightleftarrows 2\,NH_3 \quad \Delta H_r^0[298.15\,K] = -92.44\frac{KJ}{mol}\ \#(1)$$

The reaction is carried out in two catalytic reactors arranged in series. The required nitrogen comes from process air, which enters the secondary reforming reactor. The hydrogen is produced through natural gas steam reforming in two distinct reforming stages. The first stage (primary reformer) is a vertical, side fired, proprietary reactor. The second stage (secondary reformer) is an autothermal adiabatic reactor. Typical temperature and pressure of the gas stream leaving the reforming section are 1000 °C and 25 - 40 bar, respectively (Jennings, 1991). The catalysts used in the reforming reactors and in the downstream sections are sensitive to sulfur compounds (Aika et al., 1995). To avoid catalyst deactivation and poisoning, sulfur compounds are removed from natural gas in two reactors arranged in series. Similarly, carbon oxides must be removed because they are poisonous to the catalyst (Aika et al., 1995). For this reason, carbon monoxide is converted into carbon dioxide in two Water-Gas Shift reactors. Carbon dioxide is then removed in an absorption column where a Vetrocoke solution is used as a solvent (Giammarco and Giammarco, 1973). Finally, the residual amount of carbon oxides is removed in a Methanation reactor. The process stream leaving the methanator, which has the required purity for ammonia production, is compressed and sent to the ammonia synthesis loop, where ammonia is synthesized and liquefied through

subsequent cooling and expansion units. Due to thermodynamic and kinetic constraints, the ammonia synthesis has low single-pass conversion (Jennings, 1991). Therefore, part of the gases released during the liquefaction process, which consists mainly of unreacted compounds, are recycled back to the reactors.

3.1. The alarm database

The alarm database of the ammonia plant contains alarm events collected between July 2017 and November 2017. In total, 26 473 alarm events (rows of the database) occurred during the observation period. Each event is described by a set of 39 attributes (columns of the database).

Alarms are not evenly spread over the observation period. The alarm daily annunciation rate is shown in Figure 2.

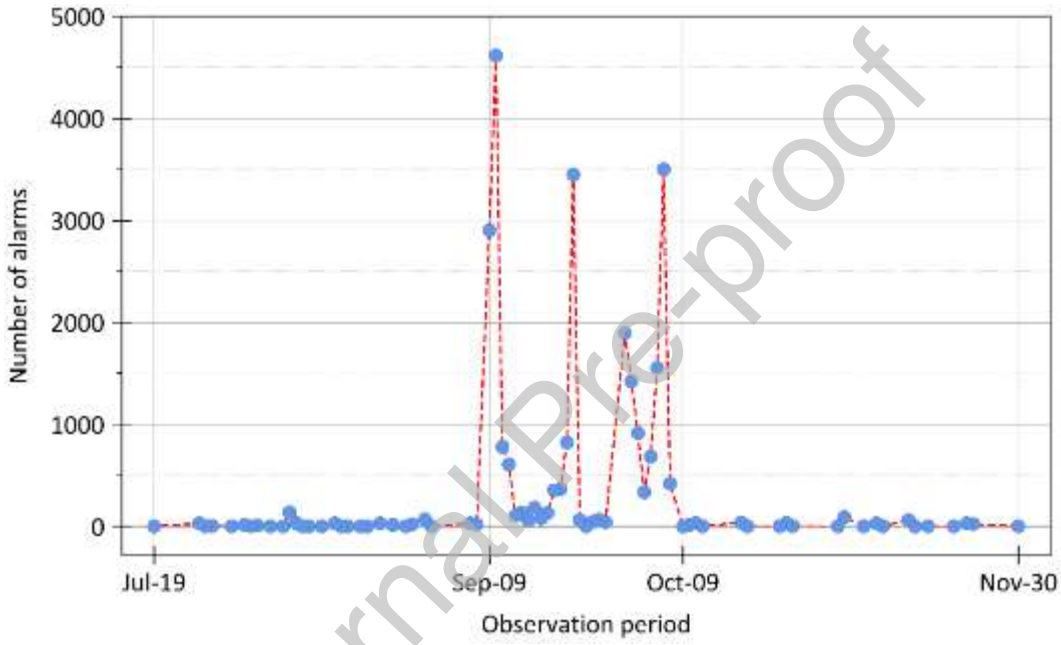

Figure 2 – Alarms' daily annunciation rate. Each circle represents the number of alarms that occurred during one day.

Over 96 % of the alarm events included in the database occurred between September and October 2017. The unusually high alarm rate was caused by a total power outage, which led to an unintended plant shut down. The plant instability and the abnormal alarm annunciation rate persisted over one month after the blackout, due to the emergency shutdown and the subsequent startup procedure. During the event, a significant number of alarm floods occurred. Therefore, the analyses described in this work have focused on that specific time-lapse (September 9th to October 9th).

Over the period of concern, 189 alarm sources produced a total of 25572 alarms. More than 72 % of them were triggered by ten alarm sources only, as shown in Figure 3.

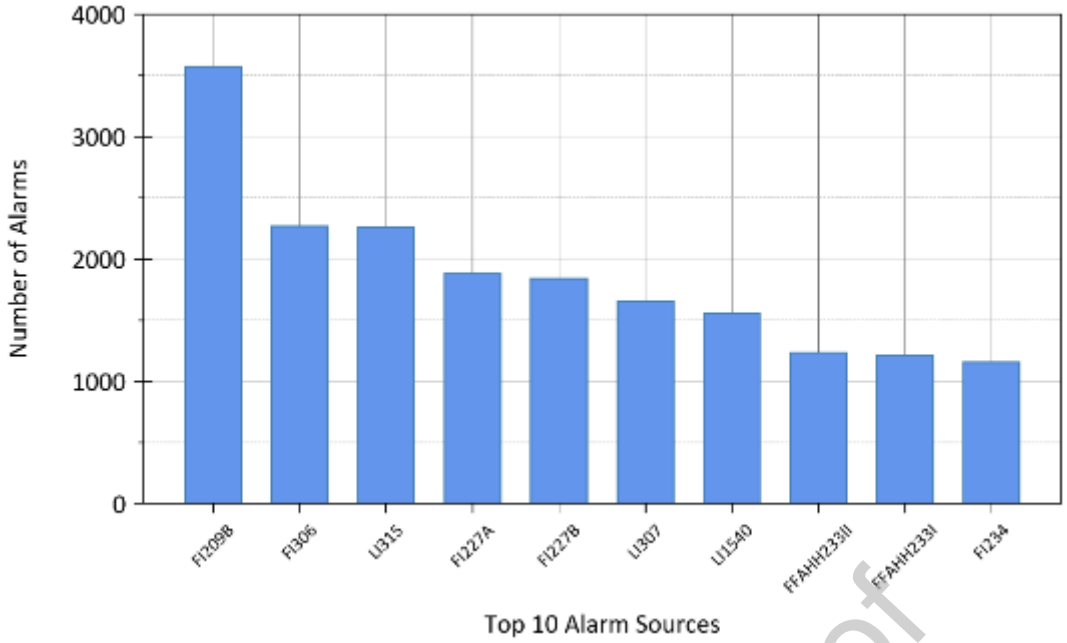

Figure 3 – The ten alarm sources with the larger alarm count over the observation period (September – October 2017).

4. Methodology

The approach follows the steps depicted in Figure 4.

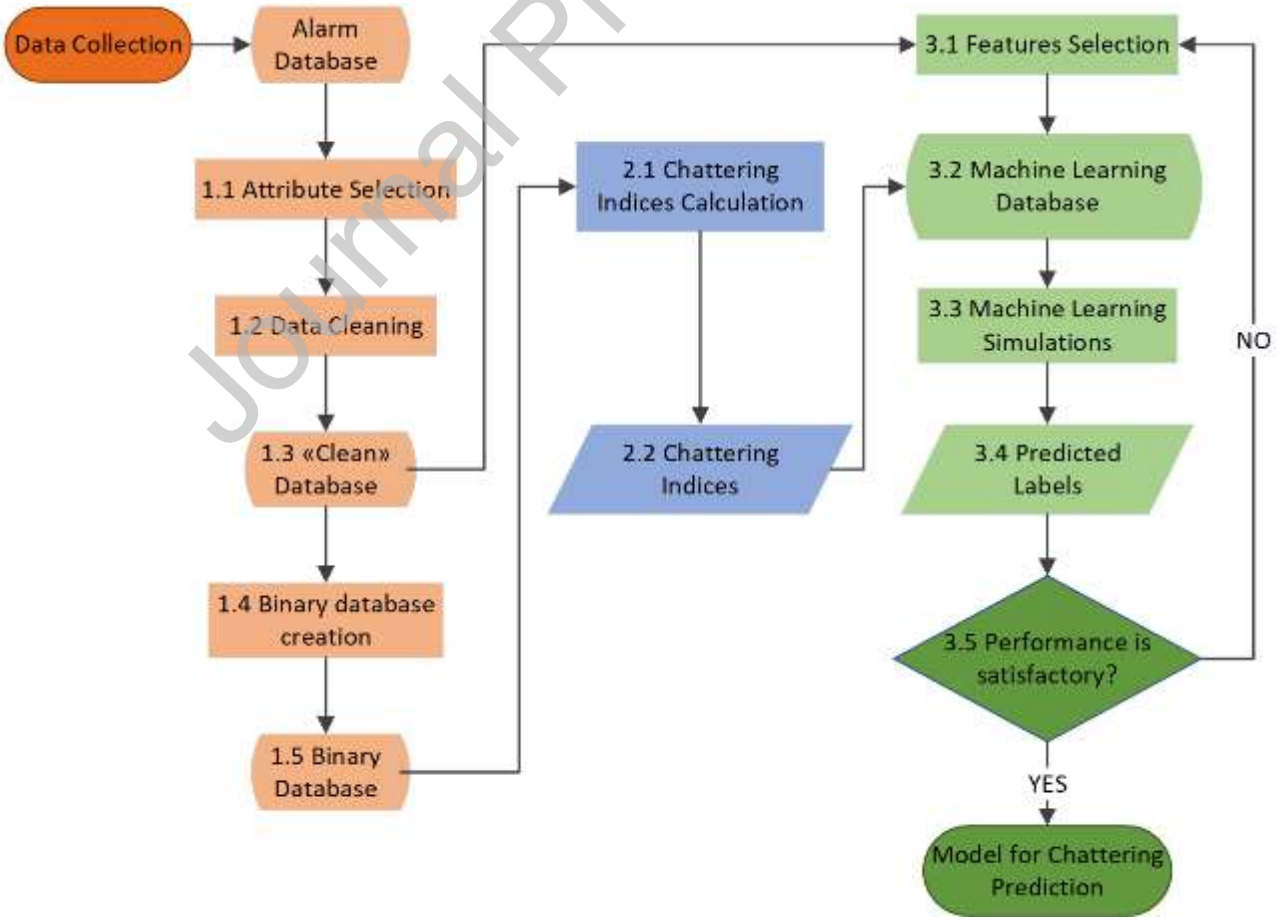

Figure 4 – Workflow of the analysis carried out. Colors represent three main stages. Stage 1 (orange): Data preprocess, Stage 2 (blue): Dynamic Chattering Indices calculation, Stage 3 (green): Machine Learning simulations. Each stage is divided into several steps, which are arranged chronologically and identified by two numbers (e.g., 1.1, 2.2, 3.5).

4.1. Data preprocessing

Raw alarm data must be prepared for the analysis.

4.1.1. Attribute selection and data cleaning

The columns of the database that are either empty or not useful for the analysis have been removed (step 1.1 in Figure 4). For example, the column "PlantHierarchy" contains standardized codes that refer to a specific plant inside the production site. The column has been removed because every alarm considered in this study comes from the ammonia production plant.

Machine Learning algorithms cannot process null (missing) values. For this reason, columns including null values were further analyzed, and, when relevant, null values were substituted by specific input values. Several techniques exist to impute missing values (Hastie et al., 2009). If the value is relevant for the analysis, one may decide to replace it with the mean or median of the non-missing values (Brink et al., 2016). If the missing value is not relevant, or if there is no way to guess the value through statistical calculations, it might be replaced with a user-defined global constant (Han et al., 2012). For example, the column "Eng. Unit" (see Table 1) contains a considerable amount of missing value due to alarms that are not associated with a measuring instrument (e.g. alarm generated by ad-hoc logics). Therefore, it has been decided to replace the missing values in the column with the symbol "– " (step 1.2 in Figure 4).

As a result, a "clean" database is obtained (step 1.3 in Figure 4), which contains only meaningful attributes and no missing values.

4.1.2. Binary Database creation

Unique alarms (i.e. the unique combination of an alarm source and an identifier) have been represented as binary sequences (step 1.4 in Figure 4). According to Kondaveeti *et al.* (2010), the binary representation of a unique alarm is an array whose elements represent one-second-spaced time bins. For a one-month-long observation period, the array has 2592000 elements (i.e., seconds in one month). The value of an element of the array can be either "1" or "0". A "1" in the sequence indicates that the unique alarm occurred at that very moment. On the contrary, a "0" means that the unique alarm did not happen. In this way, alarm occurrences are represented as 1's in the array. Finally, binary sequences are grouped in a matrix (step 1.5 in Figure 4). Rows containing zeroes only can be safely removed by the Binary Database (Kondaveeti et al., 2010).

Although it is not compulsory, representing alarm data as binary sequences will greatly simplify the calculation of the Dynamic Chattering Index.

4.2. The Dynamic Chattering Index

The mathematical formulation of the Dynamic Chattering is based on the method described by Kondaveeti *et al.* (2013) for the calculation of the Chattering Index. Both indices rely on the run-lengths distribution to quantify alarm chatter. A run-length is the "time difference in seconds between two consecutive alarms on the same tag" (Kondaveeti et al., 2013). As an example, the run-lengths related to a fictitious unique alarm are shown in Figure 5.

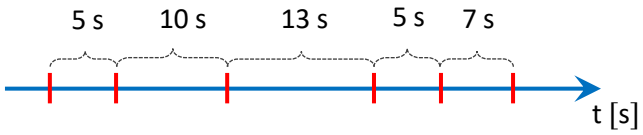

Figure 5 – Schematic representation of a unique alarm. Red sticks represent alarm events. Seconds between two subsequent sticks represent a run-length.

In the figure, the red marks represent an alarm event (i.e., a "1" in the binary sequence), the number of seconds between two consecutive marks (i.e., the time between two subsequent "1's" in the binary sequence) represent a run-length. Intuitively, considering the whole observation period, if a unique alarm has a high number of short run-lengths, it is highly probable that the alarm has shown chattering. The alarm sequence described in Figure 5 indicates Chattering because the alarm has occurred six times in less than 30 seconds. The Chattering Index in Kondaveeti *et al.* (2013) is

$$\psi = \sum_{r \in \mathbb{N}} P_r \frac{1}{r} \quad \#(2)$$

Where

- r is a natural number that represents a run-length [s] (e.g. r = 5, 7, 10, 13 for alarm data in Figure 5);
- P_r represents the probability that a run length is equal to r seconds, i.e.,

$$P_r = \frac{n_r}{\sum_{r \in N} n_r} \quad \#(3)$$

Where

- n_r is the number of run lengths equal to r seconds (e.g. $n_5 = 2$ and $n_7 = n_{10} = n_{13} = 1$ for alarm data in Figure 5);
- $\sum_{r \in \mathbb{N}} n_r$ represents the total number of run-lengths, which is one less than the unique alarm's occurrences over the observation period (e.g., the alarm in Figure 5 occurred 6 times, the summation is equal to 5).

The Chattering Index indicates the mean frequency of annunciation of a unique alarm $\left(\text{units of } \psi \text{ are } \frac{\text{alarms}}{\text{s}}\right)$, and it assumes a value between 0 and 1. A threshold value is needed to assess whether a unique alarm has shown chattering during the observation period. Kondaveeti *et al.* (2013) propose a threshold value equal to 0.05 (i.e., $\psi \geq 0.05$ indicates alarm chatter). For instance, considering

the example presented in Figure 5, the probabilities (eq.3) are $P_5 = {}^{2}/_{5}$ and $P_7 = P_{10} = P_{13} = {}^{1}/_{5}$. The Chattering Index (eq.2) is:

$$\psi = \frac{2}{5} \cdot \frac{1}{2} + \frac{1}{5} \cdot \left(\frac{1}{7} + \frac{1}{10} + \frac{1}{13}\right) = 0.26 \geq 0.05 \ \#(4)$$

which confirms that the alarm has shown Chattering behavior.

In fact, once the observation period is defined, a single ψ is obtained for each unique alarm. Although meaningful, the index is relatively static: observing the Chattering Index, one can determine whether the unique alarm has shown Chattering, but no further conclusion can be drawn (e.g., when exactly the alarm has shown Chattering). To overcome this limitation, the Chattering Index approach has been modified, and the Dynamic Chattering Index has been developed. The core idea is to calculate a regular Chattering Index every time a unique alarm occurs (i.e., every time a "1" is found in the binary representation of the alarm). Another key feature is that the calculations of the Dynamic Chattering Index involve only alarm events that occurred up to one hour after the alarm event of concern. If a unique alarm has *n* values equal to 1 in the binary sequence, *n-1* Dynamic Chattering Indices are calculated (the last event is excluded from the calculation). By this procedure, each "1" in the binary sequence is associated with a Dynamic Chattering Index, which quantifies the amount of chatter that the alarm has shown during the following hour.

Considering a generic alarm event with index *i*, the calculation of the Dynamic Chattering Index involves four steps (step 2.1 in Figure 4).

1. The largest index *k* that meets the condition $\left(Timestamp_k - Timestamp_i\right) \leq 1\ h$ is selected, and the binary sequence is reduced in such a way that only alarm events having index $j \in [i, k]$ are taken.
2. The run-lengths (r) and the number of run-lengths (n_r) of the reduced binary sequence are calculated. It is worth noting that as long as the reduced binary sequence does not include the last alarm event (i.e., if $k < n$), one run-length can be obtained for each of the alarms in the reduced sequence (i.e., the last element of the sequence is also included in the run-lengths calculations).
3. Probabilities are calculated according to equation (3).
4. The Dynamic Chattering Index of the alarm event is calculated according to equation (2).

The steps presented above are repeated $\forall\ i \in [\ 0, n-1]$ to obtain the *n* - 1 Dynamic Chattering Indices of the unique alarm of concern. Finally, the procedure is repeated for each of the unique alarms in the Binary Database.

The same threshold value discussed above has been used for alarm classification. If an alarm event has $\psi_D \geq 0.05$, the unique alarm will show chattering within one hour.

Eventually, a Dynamic Chattering Index has been calculated for each alarm event (step 2.2 in Figure 4). The use of a threshold allows classifying alarms into two categories, "Chattering within one hour" and "Not

Chattering within one hour". This result has been used to train and evaluate the Machine Learning algorithms that will be described in the following section.

4.3. Machine Learning

Machine Learning (ML) can be defined as "computational methods using experience to improve performance or to make accurate predictions" (Mohri et al., 2012). Due to the ever-increasing computation capabilities of modern calculators and to the development of computer technologies, the number of ML algorithms and their applications have witnessed extraordinary growth during the last few years (Liu et al., 2018). Despite the immense number of different algorithms, there are only three categories of ML methods, which are: Supervised Learning, Unsupervised Learning, and Reinforcement Learning. Within the present work, Binary Classification algorithms have been used, which fall into the Supervised Learning category.

A Classification algorithm takes as an input a list of *features* (i.e., meaningful attributes) of the object that must be classified and returns a *label* (i.e., the class of the object). For instance, these algorithms are employed to classify emails into "Spam" and "Not Spam" while, in the present study, the algorithm aims to classify alarm events into "Chattering within one hour" or "No Chattering within one hour". If the objects are classified into two classes only, the problem is called Binary Classification.

The selection of the most relevant features is a crucial step, and it may significantly affect the performance of the algorithm. The selection of the set of features that best represent the problem under assessment is mostly guided by experience, and a trial and error approach is often required (Brink et al., 2016)(step 3.5 in Figure 4).

The Machine Learning Classification workflow is presented in Figure 6. Two distinct stages are necessary to build and test the algorithm: *Training* and *Evaluation*.

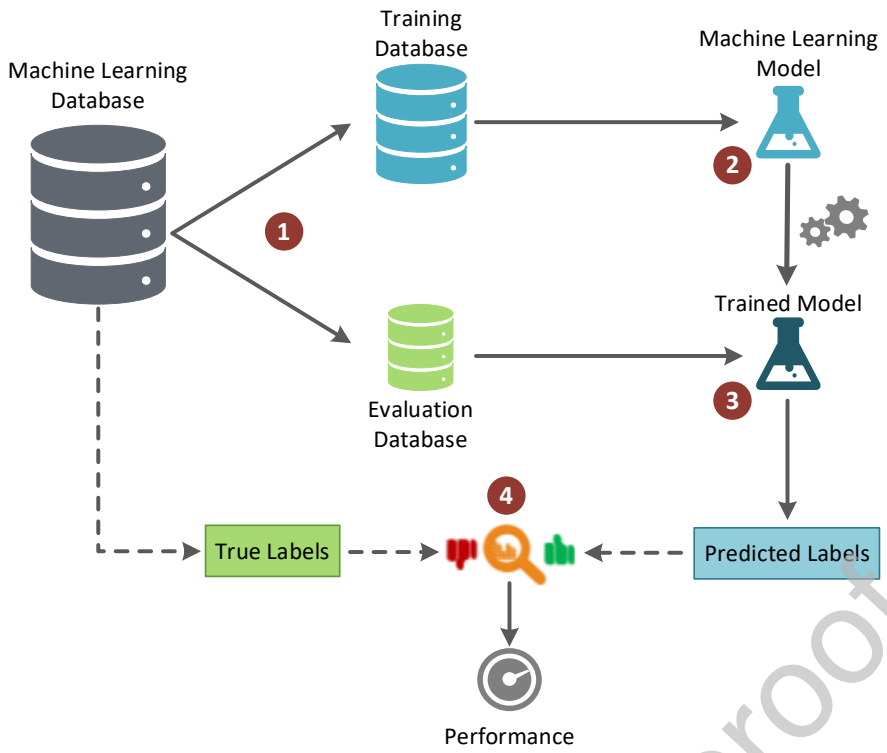

Figure 6 - Binary Classification Workflow. 1 - the database is divided into Training and Evaluation Databases. 2 - Training Database is fed to the ML model; a trained model is obtained. 3 - Evaluation Database is fed to the trained model, which predicts the labels. 4 - performance metrics are calculated.

During the training stage (step 2 in Figure 6), the algorithm receives a set of *examples*. An example is a list of features (e.g., the attributes of an alarm event) and the related label. From the examples, the algorithm "learns" the relation between features (Y) and labels (X) by optimizing the weights of an internal function (*f*).

$$Y = f\,(X) \#(5)$$

The weights are adjusted by an optimization algorithm, which aims to minimize the "distance" between *f* (X) and Y. Different types of functions exist, as well as different optimization methods.

Later, during the evaluation phase (step 3 in Figure 6), a new series of unlabeled examples (i.e., only features) are fed to the trained algorithm, which predicts the labels. Finally, the performance of the algorithm is quantified by comparing predicted labels with true labels (step 4 in Figure 6).

It is worth mentioning that the raw output of a Classification algorithm is not a label, but the label's probability (Brink et al., 2016). For example, the algorithm used in this work returns the likelihood of a unique alarm being "Chattering within one hour" or "Not chattering within one hour". A threshold is needed to convert probabilities into the final label, which is 0.5 by default (i.e., if the "Chattering within one hour" label's probability is ≥ 0.5, the model will label the alarm as "Chattering within one hour"). The *probability threshold* is an adjustable parameter, and it can significantly affect the model's performance (Google, 2020a).

4.3.1. Models

A model can be defined as "a function with learnable parameters that maps an input to an output. The optimal parameters are obtained by training the model on data. A well-trained model will provide an accurate mapping from the input to the desired output" (TensorFlow.org, 2020a). Basically, the model defines the mathematical structure of the function *f* in equation (5). Three different models have been used in this study: a Linear model, a Deep Neural Network, and a Wide&Deep model.

4.3.1.1. Linear Model

Linear models represent the labels as a linear combination of the features (Hastie et al., 2009).

$$Y = \beta_0 + \sum_{j=1}^{p} X_j \beta_j \ \#(6)$$

where:

- Y = labels;
- X = $[X_1, X_2, \dots, X_p]$ = the features vector;
- X_j = a feature;
- β_0 = intercept (or bias);
- β_j = coefficient (or weight).

In this representation, each feature has its own weight. Therefore, the model can assess how much a feature weights on the calculation of the label, but it cannot quantify the influence of combinations of features. This limitation is partially solved by crossing two or more features to create a new, more meaningful, synthetic feature (Google, 2020b). Despite that, the linear model lacks in generalization, and it cannot interpret the combination of features that never occurred during the training phase (Cheng et al., 2016).

Although simple, the model is widely used (James et al., 2013) because it is robust, fast, and performs well on large datasets. Furthermore, the weights' values are easily accessible, allowing the user to evaluate which features are more meaningful for the problem under assessment (Brink et al., 2016).

The model employed in this work uses FTRL algorithm as an optimizer (TensorFlow.org, 2020b).

4.3.1.2. Deep Neural Network

Deep Neural Networks consist of interconnected *layers*. The first layer of the network is the vector of the features (X), and the last layer is the vector of the labels (Y). Between the first and the last layer there are the so-called *hidden layers* (H). Each hidden layer is made of a certain number of *hidden units* (Z). The number of hidden layers and hidden units is a design parameter that can greatly affect the performance of the algorithm. Generally speaking, it is better to use a large number of hidden layers and units. As a drawback, bigger networks require more computational effort than networks with few layers. The model

used in this work has three hidden layers, with 1024, 512 and 256 hidden units, respectively. A schematic representation of a Neural Network is shown in Figure 7.

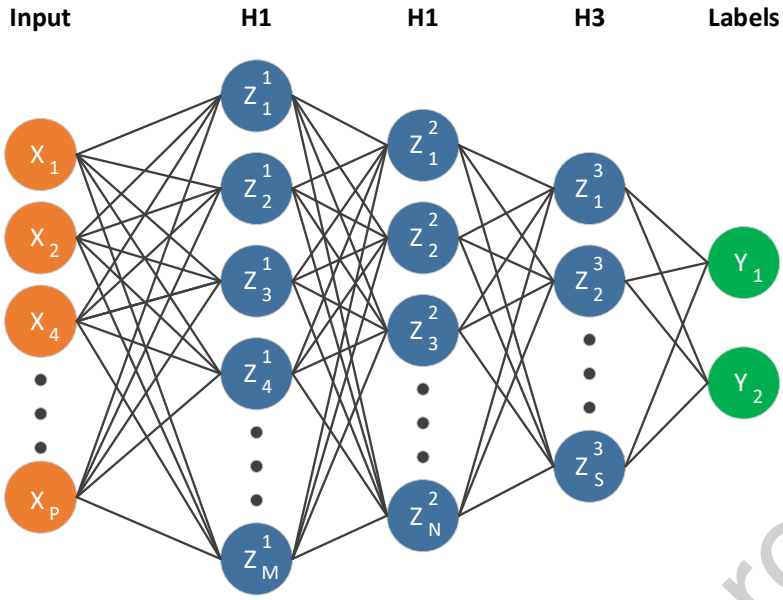

Figure 7 – Deep Neural Network with P features (orange circles), and three hidden layers (H1, H2, H3), which contain M, N, and S hidden units, respectively. The output layer (green circles) contains two labels (Y_1 and Y_2).

The connections (i.e., solid lines) in Figure 7 represent non-linear transformations. For example, the hidden units of the first and second hidden layers can be calculated as follows

$$Z_i^1 = \sigma(\alpha_{0i} + \alpha_i^T X) \qquad i = 1, \dots, M \#(7)$$

$$Z_i^2 = \sigma(\gamma_{0i} + \gamma_i^T Z^1) \qquad i = 1, \dots, N \#(8)$$

Where:

- α_{0i}, γ_{0j} = biases;
- α_i, γ_i = vectors of model coefficients;
- Z_i^k = the i-th hidden unit of the k-th hidden layer;
- $Z^1 = [Z_1^1, Z_2^1, Z_3^1, \dots, Z_M^1]$;
- σ = activation function.

Biases and coefficients are optimized during the training of the algorithm. The model employed in this work uses Adagrad algorithm as an optimizer (TensorFlow.org, 2020c), and the activation function is the linear rectifier (TensorFlow.org, 2020d).

$$\sigma(x) = \max(0, x) \#(9)$$

According to equations (7) and (8), the activation function converts the linearly combined units of a layer into the hidden units of the following layer; this allows the model to capture non-linear inter-features relationships and strengthen its generalization capabilities.

Deep Neural Networks represent state-of-the-art algorithms for audio-video processing (i.e., speech and image recognition) (Brink et al., 2016; Hastie et al., 2009) and their applications are rapidly spreading among different sectors. Although flexible, these models may over-generalize and detect non-existent relationships between features. Furthermore, they are harder to optimize compared with simpler models (e.g., the linear model).

4.3.1.3. Wide and Deep model

In an attempt to overcome the limitations of the models discussed above, Cheng *et al.* (2016) proposed a hybrid model, which is composed of a Wide part (i.e., linear) and a Deep part (i.e., Deep Neural Network), as shown in Figure 8.

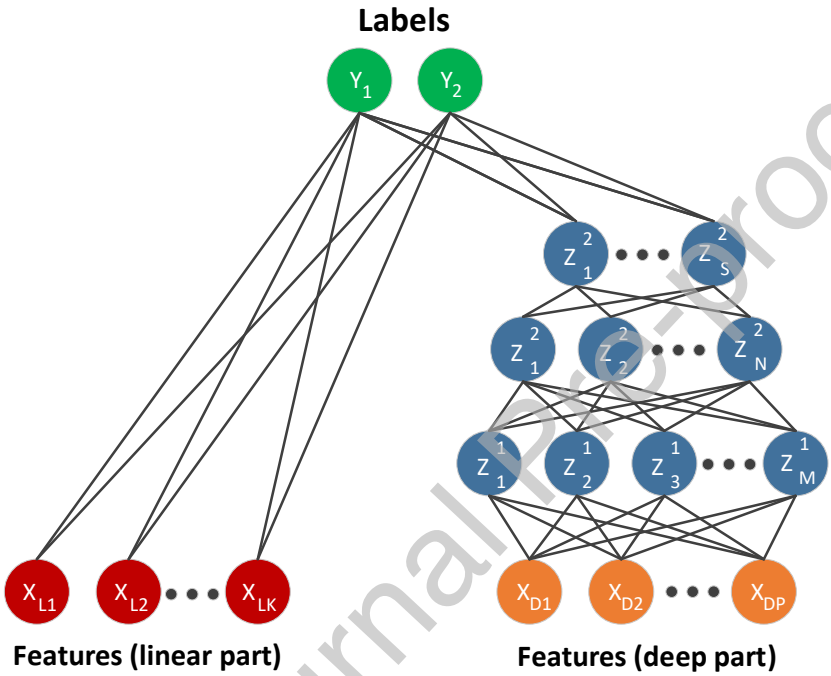

Figure 8 - Wide&Deep model, made of a linear (left) and a Deep (right) parts. The linear part takes K features (red circles). The Deep part is made of 3 hidden layers (blue circles) with M, N, and S hidden units, and takes P features (orange circles). The output layer (green circles) contains two labels (Y_1 and Y_2).

During the training phase, the parameters of the linear and deep parts are optimized simultaneously using FTRL and Adagrad algorithms. The linear part of the model could comprise both raw features and crossed-features; in this work, only crossed-features are used. The hybrid model has proven to combine the advantages of the linear model (e.g., robustness, memorization capability) and the Deep model (e.g., generalization, flexibility) minimizing their drawbacks (Cheng et al., 2016).

4.3.2. Performance indicators

The performance of a classification algorithm can be assessed by comparing predicted labels and true labels. For concision purposes, the label “No Chattering within one hour” will be referred to as the label “N”, while “Chattering within one hour” will be referred to as the label “Y”. Three metrics have been used to assess the performance

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \#(10)$$

$$Precision = \frac{TP}{TP + FP} \#(11)$$

$$Recall = \frac{TP}{TP + FN} \#(12)$$

Where

- TP = True Positive –i.e. predicted label = Y, true label = Y;
- TN = True Negative –i.e. predicted label = N, true label = N;
- FP = False Positive –i.e. predicted label = Y, true label = N;
- FN = False Negative –i.e. predicted label = N, true label = Y.

The summation of TP and TN represents the number of correct predictions and the summation of FN and FP is the number of wrong predictions. The *Accuracy* is the number of correct predictions divided by the total number of predictions, the *Precision* is the fraction of correct positive predictions, and the *Recall* is the fraction of real positive correctly predicted; the metrics assume values between 0 and 1; the larger the value, the better the metric.

As it has already been discussed, Machine Learning algorithms use a probability threshold to determine the predicted label. Therefore, changing the probability threshold can greatly affect the algorithm's performance as it modifies the values of TP, TN, FP, and FN. Unfortunately, Precision and Recall are often in tension (Google, 2020a), changes in the threshold that aim to increase the Precision may cause the Recall to decrease, and vice versa.

It is worth noting that all the metrics discussed above must be considered to evaluate the performance of a Machine Learning algorithm (Google, 2020c). A high *Accuracy* alone is meaningless and does not necessarily indicate good performances. In this work, "legitimate" alarms (i.e., that are not going to show chattering) must not be labeled as chattering ones. Therefore, the Precision is the metric that must be optimized.

5. Simulations

Three simulations have been performed, one for each model described in section 4.3.1. The Machine Learning algorithms have been built using TensorFlow r1.15 (TensorFlow.org, 2020e) running on Python 3.7.4 (Python.org, 2019). The first step to build the Machine Learning algorithms is the feature selection (step 3.1 in Figure 4). A preliminary screening has already been performed during "Attribute selection and data cleaning" (section 11) when the not useful columns have been removed from the raw alarm database.

Still, there is no guarantee that the algorithms will perform better if all columns of the "clean" database are used as features. As previously argued, feature selection often requires a trial and error approach. Different features have been tested. The best set (i.e., the one that has generated the best performance) is presented in Table 2, which contains the name and description of each feature.

Feature	Description	Format	
		Linear	Deep
Y	Year of the alarm event	Num.	Num.
M	Minute of the alarm event	Num.	Num.
D	Day of the alarm event	Num.	Num.
H	Hour of the alarm event	Num.	Num.
m	Minute of the alarm event	Num.	Num.
S	Seconds of the alarm event	Num.	Num.
SO	Source (see Table 1)	Categ	Dense
ID	Identifier (see Table 1)	Categ.	Dense
CN	Condition Name (see section 2.2)	Categ	Dense
JX	Alarm Safety function (see Table 1)	Categ.	Dense
ATD	Active Time Delta (see section 2.2)	Num.	Num.
VAL	Data Value (see Table 1)	Num.	Num.
UNI	Eng. Unit (see Table 1)	Categ.	Dense

Table 2 – Alarm features used in the simulations. Features names have been coded for concision purposes. Codes are represented in the first column and described in the second column. The last two columns represent the features format used in the linear and Deep models.

After features selection, the alarm database has been re-organized and converted into a new database (step 3.2 in Figure 4), which contains only the features listed in Table 2. Each row of the new database represents an alarm event, each of the first thirteen columns represents a feature, and the last column contains the labels. A label can be either "Y" (if $\psi_D \geq 0.05$ –i.e., the unique alarm will show chattering within one hour) or "N" (if $\psi_D < 0.05$ –i.e., the unique alarm will not show chattering within one hour).

Next, the database has been divided into two sections, to obtain the *training database* and the *evaluation database* (step 1 in Figure 6). The first part, which contains ¾ of the alarm data, has been used to train the models, and the second part (¼ of the database) to evaluate them. The last column of the evaluation database has been removed and stored as a separate variable. This prevents the model from gaining access

to the actual labels during the evaluation. Additionally, since chattering alarms are not spread evenly throughout the original database, the database has been shuffled before the division (i.e., rows have been randomly rearranged).

Prior to starting the simulations, one must ensure that the features are fed to the algorithm in a proper format. While Machine Learning models accept numerical features as an input, strings of characters (e.g., the features *Identifier*, *Source*, and *Eng. Unit* in Table 1) cannot be fed directly into the model and must be converted into a categorical or dense format. Non-numerical features that are fed to the linear model have been converted into categorical features (TensorFlow.org, 2020f). On the contrary, Deep Neural Networks do not accept categorical features as an input. Therefore, non-numerical features have been mapped into dense features (TensorFlow.org, 2020g). The last two columns of Table 2 summarize the features' format used in the linear and deep models. Furthermore, three crossed features have been used in the linear model and in the linear part of the Wide&Deep model, which are [SO x CN x JX], [SO x CN x ID], and [VAL x UNI]. This enables the linear model to assess non-linear relationships between features. In summary, the Linear model uses three crossed features in addition to the features in Table 2. The Deep model and the deep part of the Wide&Deep model use the features in Table 2, but no crossed features (because Deep models have intrinsic generalization capabilities). The linear portion of the Wide&Deep model uses three crossed features only.

Finally, the models have been trained and evaluated, as shown in Figure 6 (step 3.3 in Figure 4). After the simulations, the algorithm provided raw label probabilities, which have been converted into predicted labels using 0.5 as a threshold (step 3.4 in Figure 4). Next, predicted labels have been compared with true labels (i.e., the labels that the model should have predicted), the number of TP, TN, FP, and FN has been calculated. Accuracy, Precision, and Recall of the model have been obtained according to equations (10), (11), and (12). Finally, the performance metrics have been calculated again using different probability thresholds in order to study the effect of this parameter on the final results.

6. Results

The number of TP, TN, FP, and FN are presented in Figure 9, which contains three confusion matrices, one for each model. The rows and columns of a confusion matrix represent the true labels and the predicted labels, respectively. From top left clockwise, the elements of a confusion matrix are the number of *true negatives*, *false positives*, *true positives*, and *false negatives*. A probability threshold equal to 0.5 has been used.

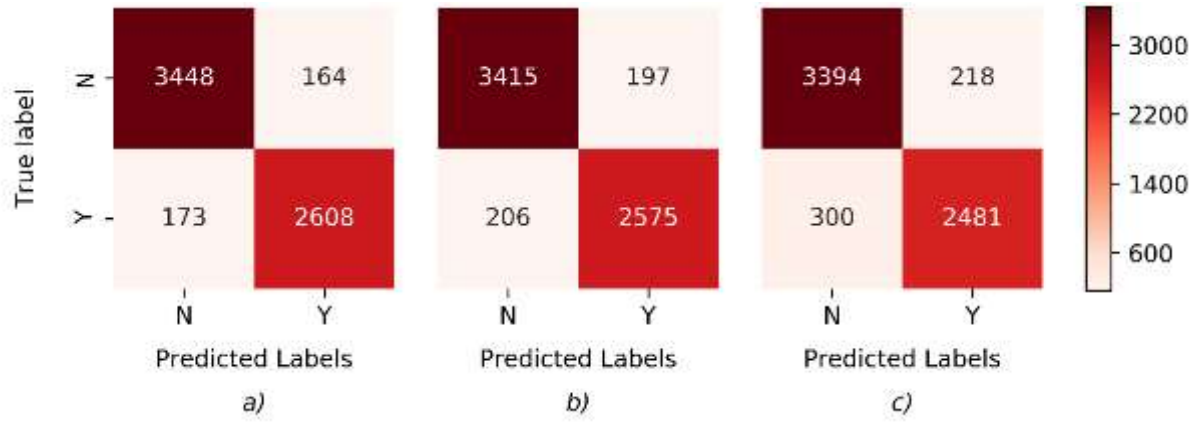

Figure 9 – Confusion matrices of the linear (a), Deep (b), and Wide&Deep (c) models. The label "N" means "No chattering within one hour", "Y" means "Chattering within one hour". TN, FP, TN, and FN are obtained using a probability threshold equal to 0.5 and color-coded according to the color bar on the right.

Metrics in Figure 9 indicate that the number of correct predictions is one order of magnitude higher than the number of wrong predictions. Moreover, the number of False Positives is always lower than the number of False Negatives. The Accuracy, Precision, and Recall achieved by each algorithm are shown in Table 3.

Model	Accuracy	Precision	Recall
Linear	0.947	0.941	0.938
Deep	0.937	0.929	0.926
Wide&Deep	0.919	0.919	0.892

Table 3 –Accuracy, Precision, and Recall achieved by the Machine Learning models. Metrics are obtained using a probability threshold equal to 0.5.

Values in Table 3 indicate that the linear model produces the largest metrics. Similarly, the Deep model achieves a better performance than the Wide&Deep model.

Figure 10, shows the Precision-Recall (P-R) curves of the three models calculated using different probability thresholds.

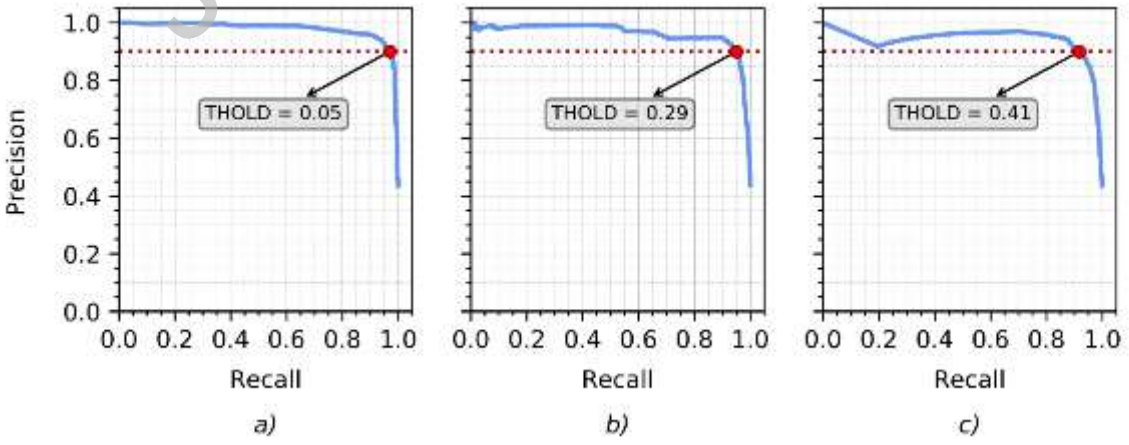

Figure 10 – Precision-Recall curves of the linear (a), Deep (b), and Wide&Deep (c) models. Probability thresholds between 0 and 1 have been used. Points of the curves represent the couple Precision – Recall at a specific threshold. Proceeding from Recall = 0 to

Recall = 1, the threshold decreases from 1 to 0 in a non-linear fashion. Red markers indicate Precision = 0.9, which is obtained at Threshold = THOLD.

As previously stated, classification algorithms provide the label probability of the events that are included in the evaluation database. A threshold value is needed to convert probabilities into labels. If the threshold is equal to 0, every alarm event in the evaluation database will be labeled as "Y". Oppositely, if the threshold is equal to 1, every alarm event in the evaluation database will be labeled as "N". Lowering the threshold causes the Recall to either decrease or to remain constant. Instead, Precision may increase or decrease when the threshold is reduced. Each point of the blue curves in Figure 10 represents the Precision and Recall values obtained using a specific threshold. For a specific model (panels a, b, and c in Figure 10), thresholds larger than THOLD ensures a Precision larger than 0.9.

7. Discussion

The Chattering Index proposed by Kondaveeti *et al.* (2013) is a valuable tool for addressing Chattering alarms retrospectively, but it does not fulfill the need for dynamicity required to achieve the objectives of the study. In fact, the Chattering Index quantifies the amount of chattering that an alarm has shown over the entire observation period: results are static, meaning that the index can be used to measure the chattering severity, but it does not provide any information about when, or why, the chattering has happened. For these reasons, the index has been modified, and a dynamic approach has been developed.

The Dynamic Chattering Index aims at quantifying the likelihood of alarm chatter after each alarm occurrence, linking past and actual process conditions to future alarm behavior. A threshold has been used to classify alarm events in two categories, "Chattering within one hour" and "Not chattering within one hour". The Dynamic version of the Chattering Index provides a more detailed picture of the alarm system performance if compared to the Chattering Index: the former classifies alarm *events,* the latter classifies *unique alarms*. In future works, the Chattering Index may be used to strengthen the Machine Learning simulations since it represents a meaningful piece of information about the past behavior of a unique alarm. For instance, one Chattering Index may be calculated for each alarm event in the database, taking into account only alarms that occurred before each event. This index may be used as a new feature in the Machine Learning simulations, allowing the model to learn the relation between past and future chattering. The approach has not been pursued in this study because the authors decided to exclude synthetic features (i.e., that requires calculation) and focus the attention on ready-to-use features (i.e., directly provided by the alarm system).

The Dynamic Chattering Index method requires to select a threshold (for alarm classification) and the length of the time interval (to obtain the reduced binary sequence, according to step 1 of the procedure described in section 4.2). In this work, a time interval equal to 1 h has been used because it appears to be a good balance between dynamicity (that cannot be achieved using large time intervals) and statistical relevance (that cannot be achieved using short time intervals). However, the choice has been arbitrary and guided by general considerations. For example, longer time intervals (e.g., 2 hours or more) may cause the index to detect Chattering even if it only appears in the last minutes of the time sequence. As a result, the index would indicate chattering for two or more hours while the alarm would not exhibit chattering for most of the time. Oppositely, shorter time intervals (e.g., 30 minutes or less) may cause the index to overestimate short run lengths and to detect chattering were no – or low – chattering exists; this issue partially affects also the index used in this study. In fact, the Dynamic Chattering Index relies strongly on statistical methods, which perform better when a large amount of data is analyzed. Unlike the Chattering Index, which considers the entire observation period, the Dynamic Chattering Index calculations involve a relatively short time interval. It may happen that the unique alarm under assessment occurred a few times during the hour, and this could lead to unexpected results. For instance, few alarm events in the reduced binary sequence will produce relatively large probabilities, since the denominator in equation (3) will be small. Besides, if some of the few run-lengths involved in the calculation are short (e.g., 1 – 10 s), the combination of short run-lengths and large probabilities will cause equation (2) to produce a large Dynamic Chattering Index, most likely higher than 0.05. As an example, consider alarm data represented in Table 4.

Index	Timestamp		Run-length [s]	ψ_D
i	09/09/2017	16:07:24	3	0.069
i + 1	09/09/2017	16:07:27	234	\
i + 2	09/09/2017	16:11:21	133	\
i + 3	09/09/2017	16:13:34	1559	\
i + 4	09/09/2017	16:39:33	2160	\
i + 5	09/09/2017	17:15:33	\	\

Table 4 – Fictitious alarm sequence. Each row represents an alarm event. The last column contains the Dynamic Chattering Index of the event i (first row the table). The symbol " \ " indicates a value that is either not calculated or not relevant for the analysis.

The calculation of the Dynamic Chattering Index of the event *i* includes all the alarms in Table 4 except the last one (because it happened later than one hour after the event *i*). Observing the run-lengths, one may conclude that the alarm did not show chattering since they appear to be long enough, and the 3 seconds long run-length alone does not seem sufficient to suggest chattering. Despite that, the calculation of the

Dynamic Chattering Index leads to an unexpected result. In particular, the run-length count and the probabilities are $n_r = 1$ and $P_r = {}^1/_5 \; \forall \, r$. Therefore, the Dynamic Chattering Index is

$$\psi_D = \frac{1}{5} \cdot \left(\frac{1}{3} + \frac{1}{234} + \cdots + \frac{1}{2160} \right) = 0.067 + 8.5 \cdot 10^{-4} + \cdots + 9.2 \cdot 10^{-5} = 0.069 \; \#(13)$$

which is greater than 0.05, and suggests chattering within one hour. Focusing on how each run-length impacts the index calculation, one can observe that a run-length equal to 3 s alone produces a contribution of 0.067, which is greater than 0.05. This behavior is due both to an extremely short run length and to large probabilities (caused by few alarms being triggered during the observation period). Usually, if many alarms occurred within the observation period, the effect of a few short run-lengths is mitigated by a small probability value. Instead, if few alarms were triggered, the probability increases, the mitigation effect stops, and an unreliably high ψ_D is produced. The issue might be avoided by excluding extremely short run-lengths or extending the time interval. The first solution has the disadvantage of ignoring extreme chattering behavior, and the second may cause loss of dynamicity. For the reasons mentioned above, future research should be devoted to the improvement of the Dynamic Chattering Index calculation in order to achieve higher reliability. For example, alarm sequences affected by the issues described above might be isolated, and different indices with different time intervals (e.g., 30 minutes, 1 hour, and 2 hours) might be tested on these sequences to assess which one performs better (i.e., which one does not overestimate the effect of few, extremely short, run lengths). Also, the Dynamic Chattering Index described in this study might be modified to take into account the number of alarm events considered in the calculation –e.g., a weighting function might be created to dampen the effects of the combination of few alarms and short run lengths. In addition,

indices calculated with different time intervals may be aggregated and used together to obtain a single, more comprehensive, and informative index.

The results of the Dynamic Chattering Index approach have been used to train three Machine Learning models for Chattering prediction. The models have shown good performances in predicting alarm chatter: results in Table 3 indicate that a high Accuracy can be achieved while maintaining high Precision. These flexible and dynamic tools may significantly improve the operators' response in different situations. During alarm floods, early warning of chattering may be delivered to the operator, who may decide to silence the alarm before it becomes a nuisance. In addition, the models could warn that the chattering is going to end (i.e., the model predicts an "N" after a sequence of "Y"), and the operator may decide to restore the alarm without the burden of checking it periodically. During normal operations, early warnings of chattering may allow the operator to investigate the issue in advance, and the ability to detect the end of a chattering sequence would prevent the alarm from being forgotten in a silenced status. In general, the models could help to increase risk awareness by providing quick and ready-to-use information and by reducing the need for manual intervention.

When the standard probability threshold is used (i.e., 0.5), the linear model qualifies as the best model since it produces the largest metrics (Table 3). Deep and Wide&Deep models show slightly smaller metrics and may need more optimization to improve their performance. On the contrary, the simpler but more robust linear model has performed better without the need of a specific optimization. The reasons why this has happened are diverse. For instance, DNN and Wide&Deep models are prone to overgeneralization and may detect inter-feature relationships where no relationship exists. The problem described in this study may need a model that is better at memorizing (e.g., Linear) rather than generalizing (e.g., DNN, Wide&Deep). Future research should investigate whether different optimization strategies (e.g., different hyperparameters, learning decay, activation functions) could improve the performance of advanced but sensitive models such as the Deep and Wide&Deep.

P-R curves in Figure 10 suggest that precisions larger than 0.9 can always be achieved while maintaining the Recall close to 0.9 by varying the probability threshold. If the threshold is further reduced (i.e., below 0.05 for the linear model, 0.29 for the Deep, and 0.41 for the Wide&Deep), the Precision drops significantly. The selection of the best threshold (i.e., threshold tuning) strongly depends on the specific problem under assessment (e.g., unbalanced/balanced dataset, cost-sensitive/insensitive classification)(Brink et al., 2016; Google, 2020d; Ling and Sheng, 2008). Misclassifying legitimate alarms (FP) is more critical than misclassifying chattering alarms (FN) as a False Positive may cause the operator to silence a legitimate alarm. Therefore, False Positives must be avoided, and Precision must be increased. Unfortunately, increasing the Precision often causes the Recall to decrease (Brink et al., 2016). The best threshold must ensure a high Precision while maintaining the Recall to an adequate level. Acceptable thresholds may be identified by selecting minimum values of Precision and Recalls, but selecting the best threshold requires more considerations. Often when classification errors have different criticality, a process similar to cost-benefit analysis is needed to identify the best threshold value (Ling and Sheng, 2008; Sheng and Ling, 2006). Other approaches involve the optimization of the weighted harmonic mean between Precision and Recall (F_β - measure)(Chai, 2005).

As a final note on thresholds and P-R curves, it is worth noting that the linear model provides Precisions greater than 0.90 when thresholds between 1 and 0.05 are used. This means that whenever the model predicts the label “1” (i.e., “Chattering within one hour”), it produces a large probability value, which is often larger than 0.95 (i.e., 1 - 0.05). In other words, the linear model is extremely “confident” when predicting chattering alarms.

Focusing on the Linear model, the nature of wrong predictions (i.e., FN and FP) has been studied more in detail. Three leading causes of error have been identified:

1. The model could not identify the beginning of a chattering series.
2. The model could not identify the end of a chattering series.
3. The model labels all the events of the unique alarm of concern as “Y” or “N”.

Cause 1 occurs when the model fails to identify the first element of a Chattering sequence or, in other words, it fails to detect the first unique alarm event labeled as “Y” after one or more events labeled as “N”. Figure 11 clarifies this insight.

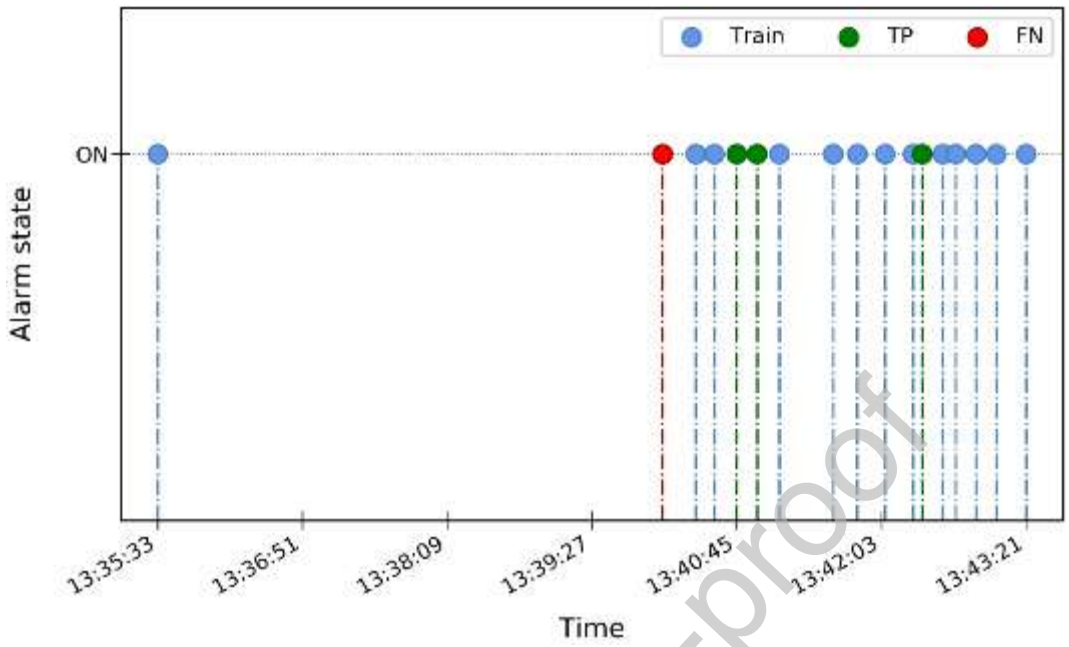

Figure 11 – Detail of a Chattering sequence produced by the unique alarm FI234 LTRP. Colored dots represent alarm events (alarm state = “ON”). True label is “Y” for all the events in the figure. Blue dots refer to alarm events included in the Training database, other colors refer to events included in the Evaluation database. Red dots indicate a wrong prediction (a False Negative), green dots indicate a correct prediction (a True Positive).

As one might notice, the first event of the chattering sequence (the red dot in Figure 11) has been incorrectly labeled (true label is Y, predicted label is N), and a False Negative has been produced as a consequence. Later in time, the model has correctly identified chattering (green dots). Also, the model has correctly predicted the end of the chattering sequence, which occurred at 13:56:00 (not displayed in Figure 11).

Cause 2 occurs when the model fails to identify the last element of a Chattering sequence. Figure 12 provides an example of this.

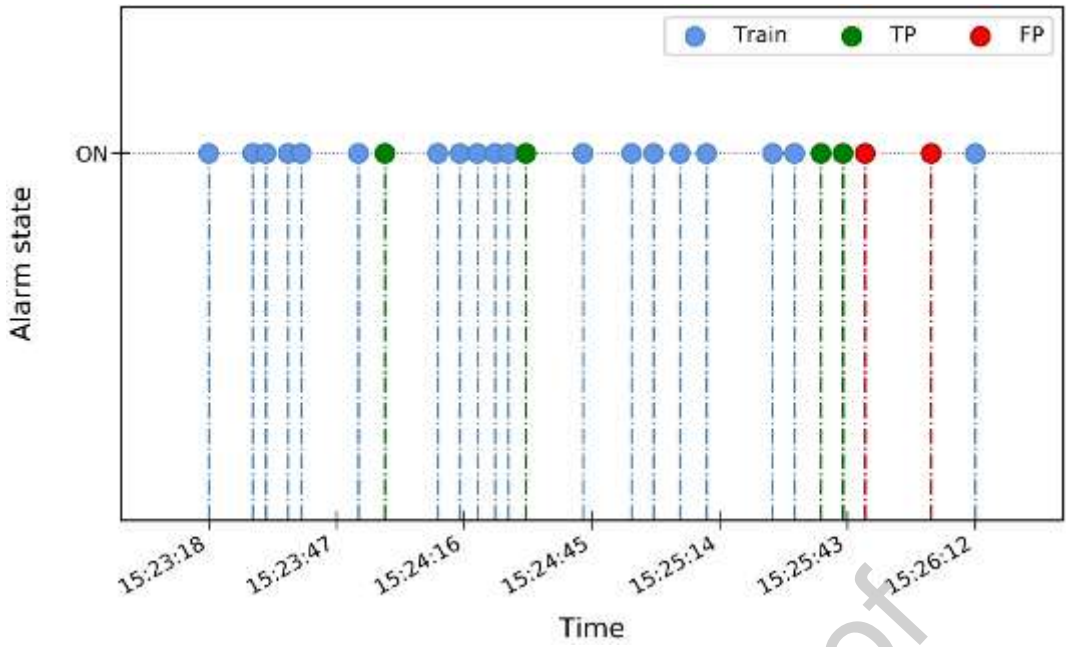

Figure 12 – Detail of a Chattering sequence produced by the unique alarm FI234 LTRP. Colored dots represent alarm events (alarm state = "ON"). True labels of the last three elements are "N", other events have "Y" as True labels. Blue dots refer to alarm events included in the Training database, other colors refer to events included in the Evaluation database. Red dots indicate a wrong prediction (a False Positive), green dots indicate a correct prediction (a True Positive).

In Figure 12, the last two unique alarm events of the series (red dots) have been incorrectly labeled (the true label is N, while the predicted label is Y), and two False Positive have been produced as a consequence. Regarding cause 3, it may happen that if the true labels related to a unique alarm are strongly unbalanced (i.e., mostly "Y" or "N"), the model will deduce that all the events produced by that particular unique alarm must be labeled as "Y" or "N", depending on which is the most frequent. For instance, this behavior has been observed for both the unique alarms "LI315 IOP" and "TI542 IOP": the first produced a total of 18 alarm events and only 5 of them were "Not Chattering within one hour", the latter produced only 4 "Chattering within one hour" events out of 38 in total. As a consequence, the algorithm has predicted that all the events produced by "LI315 IOP" must be labeled as "Y", and events produced by "TI542 IOP" must be labeled as "N".

Poor data distribution, as well as the use of too small datasets, may play a crucial role in causing the issues described above. For this reason, it might be worthwhile to consider a more extensive database for further analyses. Besides, different sets of features should be tested to resolve the misidentification of chattering sequences boundaries.

8. Conclusions

A Machine Learning method for chattering prediction was developed. Analyses have involved the formulation of the Dynamic Chattering Index to perform a preliminary classification of historical alarm data.

This new index overcomes the limitations of the Chattering Index, providing more flexible and dynamic results, which can be used to link actual process conditions to future alarm behavior.

Three different Machine Learning models –Linear, Deep, and Wide&Deep– have been trained and evaluated. The models have been tested on the ability to predict future chattering behavior based on actual process conditions. The performance metrics and the P-R curves indicate robustness and good prediction capability of the models. The method may be used to build an online tool for chattering prediction and decision-making support. For instance, the algorithm could provide early warnings of possible chattering, and actions might be taken by the operator to avoid this event. Consequently, the workload would be reduced, and the risk of alarm floods would be minimized. In general, the approach demonstrates that advanced analysis techniques can be used to extract knowledge from historical data and perform accurate predictions. A data-driven approach for process monitoring and control appears to be a valuable and interesting opportunity to exploit process data and increase process safety and stability.

9. References

Ahmed, K., Izadi, I., Chen, T., Joe, D., Burton, T., 2013. Similarity analysis of industrial alarm flood data. IEEE Trans. Autom. Sci. Eng. 10, 452–457. https://doi.org/10.1109/TASE.2012.2230627

Aika, K., Christiansen, L.J., Dybkjaer, I., Hansen, J.B., Nielsen, P.E.H., Nielsen, A., Stoltze, P., Tamaru, K., 1995. Ammonia. Springer Berlin Heidelberg, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-79197-0

Aleixandre, J., Alvarez, I., García, M., Lizama, V., 2015. Application of Multivariate Regression Methods to Predict Sensory Quality of Red Wines. Czech J. Food Sci. 33, 217–227. https://doi.org/10.17221/370/2014-CJFS

ANSI/ISA, 2016. ANSI/ISA–18.2–2016 Management of Alarm Systems for the Process Industries. ANSI/ISA.

Beebe, D., Ferrer, S., Logerot, D., 2012. Alarm floods and plant incidents 17.

Brink, H., Richards, J., Fetherolf, M., 2016. Real-World Machine Learning, first. ed. Manning Publications, Shelter Island.

Chai, K.M.A., 2005. Expectation of f-measures: Tractable exact computation and some empirical observations of its properties. SIGIR 2005 - Proc. 28th Annu. Int. ACM SIGIR Conf. Res. Dev. Inf. Retr. 593–594. https://doi.org/10.1145/1076034.1076144

Cheng, H.-T., Koc, L., Harmsen, J., Shaked, T., Chandra, T., Aradhye, H., Anderson, G., Corrado, G., Chai, W., Ispir, M., 2016. Wide & deep learning for recommender systems, in: Proceedings of the 1st Workshop on Deep Learning for Recommender Systems. pp. 7–10.

EEMUA, 2013. EEMUA Publication 191 Alarm systems - a guide to design, management and procurement.

European Union, 2012. L 197. Off. J. Eur. Union 55, 38–71. https://doi.org/10.3000/19770677.L_2012.197.eng

Ge, Z., Song, Z., Ding, S.X., Huang, B., 2017. Data Mining and Analytics in the Process Industry: The Role of Machine Learning. IEEE Access 5, 20590–20616. https://doi.org/10.1109/ACCESS.2017.2756872

Giammarco, G., Giammarco, P., 1973. Process for Eliminating CO2 and/or H2S from Gaseous Mixtures. 3725592.

Google, 2020a. Classification: Precision and Recall | Machine Learning Crash Course [WWW Document]. URL https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall (accessed 1.24.20).

Google, 2020b. Feature Crosses | Machine Learning Crash Course | Google Developers [WWW Document]. URL https://developers.google.com/machine-learning/crash-course/feature-crosses/video-lecture (accessed 1.24.20).

Google, 2020c. Classification: Accuracy | Machine Learning Crash Course [WWW Document]. URL https://developers.google.com/machine-learning/crash-course/classification/accuracy (accessed 1.24.20).

Google, 2020d. Classification: Thresholding [WWW Document]. URL https://developers.google.com/machine-learning/crash-course/classification/thresholding (accessed 6.15.20).

Han, J., Kamber, M., Pei, J., 2012. Data Mining: Concepts and Techniques, Data Mining: Concepts and Techniques. Elsevier Inc. https://doi.org/10.1016/C2009-0-61819-5

Hastie, T., Friedman, R., Tibshirani, J., 2009. The Elements of Statistical Learning. Springer-Verlag New York. https://doi.org/10.1007/978-0-387-84858-7

Health and Safety Executive, 1997. The Explosion and Fires at the Texaco Refinery, Milford Haven, 24 July 1994: A Report of the Investigation by the Health and Safety Executive Into the Explosion and Fires on the Pembroke Cracking Company Plant at the Texaco Refinery, Milford Haven on 24 J, Incident Report Series. HSE Books.

James, G., Hastie, T., Tibshirani, R., Witten, D., 2013. An Introduction to Statistical Learning: With Applications in R. Springer-Verlag New York. https://doi.org/10.1007/978-1-4614-7138-7

Jennings, J.R., 1991. Catalytic Ammonia Synthesis, Springer Science & Business Media. https://doi.org/10.1007/978-1-4757-9592-9

Katzel, J., 2007. Control Engineering | Managing Alarms [WWW Document]. URL www.controleng.com/articles/managing-alarms (accessed 1.23.20).

Kondaveeti, S.R., Izadi, I., Shah, S.L., Black, T., 2010. Graphical representation of industrial alarm data. IFAC Proc. Vol. 11, 181–186. https://doi.org/10.3182/20100831-4-fr-2021.00033

Kondaveeti, S.R., Izadi, I., Shah, S.L., Shook, D.S., Kadali, R., Chen, T., 2013. Quantification of alarm chatter based on run length distributions. Chem. Eng. Res. Des. 91, 2550–2558. https://doi.org/10.1016/j.cherd.2013.02.028

Kordic, S., Lam, C.P., Xiao, J., Li, H., 2010. Patterns Relevant to the Temporal Data-Context of an Alarm of Interest, in: Dynamic and Advanced Data Mining for Progressing Technological Development. IGI Global, pp. 18–39. https://doi.org/10.4018/978-1-60566-908-3.ch002

Laberge, J.C., Bullemer, P., Tolsma, M., Reising, D.V.C., 2014. Addressing alarm flood situations in the process industries through alarm summary display design and alarm response strategy. Int. J. Ind. Ergon. 44, 395–406. https://doi.org/10.1016/j.ergon.2013.11.008

Ling, C.X., Sheng, V.S., 2008. Cost-Sensitive Learning and the Class Imbalance Problem. Encycl. Mach. Learn. 231–235. https://doi.org/10.1.1.15.7095

Liu, J., Kong, X., Xia, F., Bai, X., Wang, L., Qing, Q., Lee, I., 2018. Artificial intelligence in the 21st century. IEEE Access 6, 34403–34421. https://doi.org/10.1109/ACCESS.2018.2819688

Mahadevan, S., Shah, S.L., 2009. Fault detection and diagnosis in process data using one-class support vector machines. J. Process Control 19, 1627–1639. https://doi.org/10.1016/j.jprocont.2009.07.011

Miao, A., Ge, Z., Song, Z., Zhou, L., 2013. Time neighborhood preserving embedding model and its application for fault detection. Ind. Eng. Chem. Res. 52, 13717–13729. https://doi.org/10.1021/ie400854f

Mohri, M., Rostamizadeh, A., Talwalkar, A., 2012. Foundations of Machine Learning, first. ed, Adaptive Computation and Machine Learning series. MIT Press, Cambridge.

Paltrinieri, N., Comfort, L., Reniers, G., 2019. Learning about risk: Machine learning for risk assessment. Saf. Sci. 118, 475–486. https://doi.org/10.1016/j.ssci.2019.06.001

Python.org, 2019. Python Release Python 3.7.4 | Python.org [WWW Document]. URL https://www.python.org/downloads/release/python-374/ (accessed 4.23.20).

Reis, M.S., Kenett, R., 2018. Assessing the value of information of data-centric activities in the chemical processing industry 4.0. AIChE J. 64, 3868–3881. https://doi.org/10.1002/aic.16203

Shaw, J.A., 1993. DCS-based alarms: Integrating traditional functions into modern technology. ISA Trans. 32, 177–181. https://doi.org/10.1016/0019-0578(93)90039-Y

Sheng, V.S., Ling, C.X., 2006. Thresholding for making classifiers cost-sensitive. Proc. Natl. Conf. Artif. Intell. 1, 476–481.

Stanton, N.A., Barber, C., 1995. Alarm-initiated activities: an analysis of alarm handlingby operators using text-based alarm systems in supervisory control systems. Ergonomics 38, 2414–2431. https://doi.org/10.1080/00140139508925276

TensorFlow.org, 2020a. Models and layers | TensorFlow.js [WWW Document]. URL https://www.tensorflow.org/js/guide/models_and_layers (accessed 1.24.20).

TensorFlow.org, 2020b. tf.keras.optimizers.Ftrl | TensorFlow Core v2.1.0 [WWW Document]. URL https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Ftrl (accessed 4.25.20).

TensorFlow.org, 2020c. tf.keras.optimizers.Adagrad | TensorFlow Core v2.1.0 [WWW Document]. URL https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adagrad (accessed 4.25.20).

TensorFlow.org, 2020d. tf.nn.relu | TensorFlow Core v2.1.0 [WWW Document]. URL https://www.tensorflow.org/api_docs/python/tf/nn/relu (accessed 4.23.20).

TensorFlow.org, 2020e. TensorFlow [WWW Document]. URL https://www.tensorflow.org/ (accessed 4.23.20).

TensorFlow.org, 2020f. tf.feature_column.categorical_column_with_vocabulary_list [WWW Document]. URL https://www.tensorflow.org/api_docs/python/tf/feature_column/categorical_column_with_vocabulary_list (accessed 4.23.20).

TensorFlow.org, 2020g. tf.feature_column.indicator_column | TensorFlow Core v2.1.0 [WWW Document]. URL https://www.tensorflow.org/api_docs/python/tf/feature_column/indicator_column (accessed 4.23.20).

Weiss, J., 2010. Protecting Industrial Control Systems from Electronic Threats, first. ed. Momentum Press, New York.

Zhang, Yingwei, Teng, Y., Zhang, Yang, 2010. Complex process quality prediction using modified kernel partial least squares. Chem. Eng. Sci. 65, 2153–2158. https://doi.org/10.1016/j.ces.2009.12.010

Zhong, S., Wen, Q., Ge, Z., 2014. Semi-supervised Fisher discriminant analysis model for fault classification in industrial processes. Chemom. Intell. Lab. Syst. 138, 203–211. https://doi.org/10.1016/j.chemolab.2014.08.008

1. Author statement

Nicola Tamascelli: conceptualization, methodology, software, validation, formal analysis, writing – original draft, writing – review and editing, visualization.

Nicola Paltrinieri: conceptualization, methodology, software, data curation, supervision, project administration, funding acquisition.

Valerio Cozzani: supervision, project administration, funding acquisition.

Highlights

- A new method for detecting and quantifying alarm chatter is proposed.
- Machine Learning is used to predict alarm chatter based on actual process conditions.
- Three Classification algorithms are trained and compared: Linear, Deep, Wide&Deep.
- Machine Learning can be used to exploit historical data and improve process safety.
- The method is presented and discussed using an industrial alarm database.

Declaration of interests

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