General Model for Metrics Calculation and Behavior Prediction in Manufacturing Industry

An Automatic Machine Learning Approach

Maria João Lopes
Bosch Termotecnologia SA, Universidade de Aveiro, Portugal

Eugénio Rocha
Universidade de Aveiro, Portugal

Pétia Georgieva
Universidade de Aveiro, Portugal

Nelson Ferreira
Bosch Termotecnologia SA, Portugal

ABSTRACT

In this chapter, a comprehensive description of a generic framework aimed at solving various predictive data-driven related use cases, occurring in the manufacturing industry, is provided. The framework is rooted in a general mathematical model so called Queue Directed Graph (QDG). With the aid of QDGs and containerized microservices implementations, the typology of the system is analyzed and real use cases are explained. The goal is for this framework to be able to be used to all use cases which fit in this typology. As an example, a data generation distribution model is proposed, the parameter stability and predictive robustness is studied, and automated machine learning approaches are discussed to predict the throughput time of products in a manufacturing production line just by knowing the processing time in their first stations.

Keywords: Industry 4.0, Manufacturing Production Line, Machine Learning, AutoML, Processing Time Distribution, Queue Directed Graph, Parameter Stability, Predictive Robustness, Classifier's Overfitting

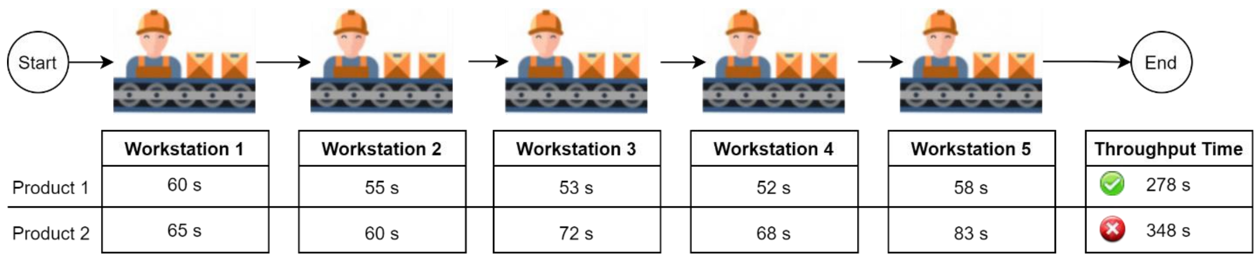
INTRODUCTION

A characteristic aim in the Manufacturing Industry is to optimize the operations in and around production lines (PL). For instance, simulation has been one of the most widely used techniques as shown in Russkikh (2020). Most of these sorts of problems, however, fit into a much broader category and with the right assumptions it is possible to create a model just generic enough to be able to solve a myriad of problems around this abstract entity, being it a PL, a value stream or a material delivery flow.

In order to understand the productive process, it is very common for organizations to resort to KPIs (Key Performance Indicators) and metrics. One of the most well-known is the *throughput time*. The throughput time describes the amount of time that it takes for a product to go through the production line. It can be calculated as the end processing minus the start processing time. Time based metrics are used extensively because they contain within other aspects of the process which would be hidden by other metrics. For example, if the throughput time exceeds some expected value, then it may reveal other underlying factors as the fact that not enough components had being fed to the PL, so the process will be starving at some point; the experience of the workers; a certain product family which is more complex to produce among others.

In Figure 1, a practical example of this calculation is featured. In this image, there is a schematical representation of a PL with a total of 5 workstations. The throughput time is calculated as the sum of all the *processing times* as duration of time the products spent in each workstation. Notice how there might be a trend already at the first 3 stations of the PL where Product 2 takes systematically more time than Product 1. This sort of trend is precisely what a prediction solution would like to catch and to see whether, from a small number of firsts stations of the PL, a significant trend in the throughput time can be inferred. With this information, one could classify the parts prematurely as efficient (such as Product 1) or non-efficient (as Product 2).

Figure 1. Schematic representation of the throughput time calculation in a production line.



	Workstation 1	Workstation 2	Workstation 3	Workstation 4	Workstation 5	Throughput Time
Product 1	60 s	55 s	53 s	52 s	58 s	✓ 278 s
Product 2	65 s	60 s	72 s	68 s	83 s	✗ 348 s

The focus of this work is to construct a binary classifier where, given a certain effectiveness threshold plus the processing time at the first N stations, i.e., a small amount compared to the total size of the line, the solution is able to predict the outcome of the throughput time class.

In order to have an approach which would fit PLs in a broader sense, and even other types of systems with a similar topology, it is necessary to propose a generic model. The example that was seen above is actually an over-simplification of such manufacturing processes. For instance, it is considered that the throughput time is simply a sum over the different processing times in each workstation. However, there are actually waiting times in between the stations because they are not always available to start processing parts since they may be processing still the previous part. For this reason, the concept of queues will also be introduced later on. Queues are entities where parts wait for their turn to be processed in a FIFO (First In, First Out) sequence.

Queues make the system more realistic yet there are other aspects to be taken into consideration such as reworks (repeating the process of a part more than once in the same station). In these scenarios, the part does not go back to the queue as this would be a waste of time for the process itself. However, this sort of changes to the usual topology must be taken into account.

A generic model allows one to calculate a myriad of different metrics which can occur at various levels. They can occur at the station level, for instance, the *processing time*, i.e., the time it take for the station to process one part. They can also occur at the transition level. One example is the *waiting time*, the time it takes for the part to end its processing in station N and to start its processing in station N+1. Finally, there are the so-called global metrics. The throughput time is an example of such a metric since it concerns the global entity, i.e., the PL. This classification is valid for metrics from the workstations point of view. There are also other types of metrics which can be calculated from the perspective of the part rather than that of the workstation. With this sort of generalization, this model would be valid not only for PLs, but for other systems which follow the same logic. These could be, for instance, trucks delivering goods or invoices flowing from supplier to customer.

The metrics described above can then either be presented through a dashboard or through an Early Warning System (EQS) which would notify the PL's Team Leaders of complications for a certain part. With this information, reactive measures could be taken immediately. The idea of doing the prediction for the throughput time metric arises precisely of the need of having enough time available to be able to take action in a meaningful timeframe. The main goal is also that the metrics can be reused and used for other higher-level applications, i.e., metrics calculation that are done on top of existing ones. This abstraction allows the calculation of other metrics in a much easier way since the baseline already contains processed information one can use.

In the next section, some use cases found in the literature will be reviewed, along with the Machine Learning (ML) techniques that can be leveraged to solve such issues. Afterwards, the focus will shift towards presenting a mathematical data model and its characteristics. Defining a precise model is quite relevant to ensure that all assumptions are fulfilled by the real data collected and that the model accounts for particular phenomena of the physical process, that in several models are simply not taken into account. Once this theoretical construct has been properly introduced, the robustness of the method will be tested through a series of experiments using simulated data. With these results as basis, the method is applied to a real PL. Subsequently, the authors dwell on the challenges of implementing such use cases in an organization and propose some suggestions to tackle them. The chapter ends with the conclusion and future research directions to be followed.

BACKGROUND

The usage of mathematical methods and Artificial Intelligence (AI) is becoming an increasingly studied subject in the academia in recent years. However, it can be quite challenging for executive and upper-level management to pinpoint actual usages which can translate this premise into actual value for their organizations. A more detailed look into the challenges that these entities are facing can be found near the end of the chapter.

Columbus (2019) presents different examples of the utilization of ML techniques to the added benefit of manufacturing corporations. These range from supply chain, to process or product design optimization. Even more qualitative areas such as Human Resources can benefit tremendously from these applications.

However, the real adoption rate is still reduced. According to Lee (2020), there are few articles that focus on direct applications on ML techniques at the enterprise level. The ones available focus mainly in the banking and finance sector. These are the sectors that are more heavily invested in these technologies. In industrial settings, the most widely known applications are generally reserved to the tuning of process parameters, as discussed in Chen (2017) and Weichert (2019). These can be applied in a wide range of processes such as Welding or Plastic Injection Molding. Still, the overall scenario in the entrepreneurial realm is that there are few actual deployments and most of them are still in a pilot phase.

Another important aspect mentioned in the literature is the process of algorithm selection to solve a concrete problem. This is an aspect the authors are deeply interested in tackling and one reason why the development of the framework at hand is being done using an agile approach. This way one can clearly define the characteristics that a problem must have in order to fit in a certain category. This is a widely known issue, that there is not one single algorithm that proves to be a good fit for all problems. This limitation is explored extensively in Pedro Domingos' book, The Master Algorithm (Domingos, 2017).

With this context in mind, one can still find some examples in the recent literature which address some areas which will be in focus during this chapter. One of them are Random Forests classifiers which are explored in Wu (2017). In this article, the authors compare the performance of Artificial Neural Networks, Support Vector Regression and Random Forests since in previous literature, there was a good performance of decision trees which are closely related method to Random Forests. Using Random Forests proved a reliable solution according to the author’s conclusion. The method was able to predict accurately the wear of the tool in milling processes. Another completely different application of this algorithm will be explored in this work.

In more specific terms, this chapter covers the usage of a graph based, generic approach for solving a number of different problems. Here, the approach will be focused in the prediction of the throughput time. In Backus (2006), the focus was on predicting the cycle time of intermediate cycle times in the semi-conductor industry. One of the issues the researchers ran into was the influence of outliers in the predictive model. For this reason, the batches which exhibited a deviating behavior were excluded from the scope. The authors concluded that a real time application would be a desirable next step. These two questions, how to treat the outliers and how to use the model in real time are important focus points in this work.

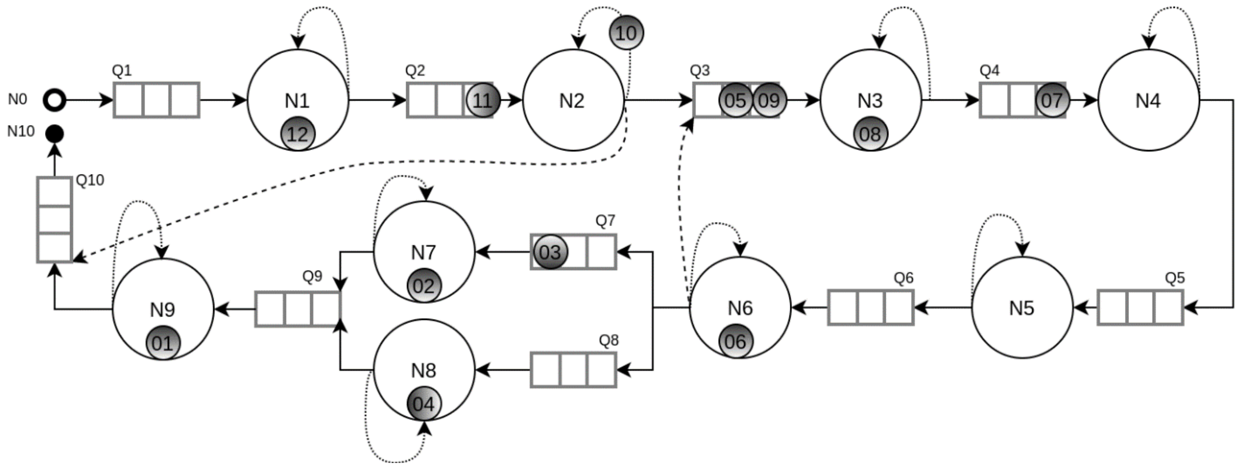
This study will also cover the usage of Automated ML (AutoML) as a way of achieving high performance results. In Tušar (2017), overfitting is referred and conclusions state that it can be reduced by using tree ensembles, giving as an example the Random Forests model. However, AutoML is known for being intrinsically an overfitting strategy, since a heavy number of combinations and optimizations are ran in order to achieve the best possible results. It will be assessed whether this characteristic can be used to one’s advantage later.

DATA MODELLING

A production line (PL) may be modeled in quite different ways, depending on the goal of the study. Common ways to formally describe workflow processes include Petri net variants or activity diagram variants. Even though they show similarities, fundamental differences turn their applications into PLs too complex or lacking relevant features, see Eshuis and Wieringa (2003). Several of these models assume that parts are produced or consumed by actors and do not incorporate (explicitly) the notion of layers of queues and their associated nodes, see Buck and Lee (1993).

For a better understand of the problem, a mathematical structure for modelling a graph with queues is introduced, allowing to incorporate the assumptions and metadata that is common in a PL. At its core, a graph is merely a set of nodes and arcs, which can easily be associated with physical entities (workstations, people or servers). The nodes are connected between one another by arcs, describing paths of product transitions. Nodes and arcs are the infrastructure but truly important are the products, the data that they have associated, and data operations that can be applied in each component of the graph.

Figure 2. Schematic representation of a production line, its stations and products flowing in it.



In Figure 2, nodes (identified from N_1 to N_9) represent transformation stations in the PL, tokens (small dark circles with numbers from 01 to 12) represent parts or materials moving in the PL, queues (identified by Q_1 to Q_{10}) are intermediate locations where tokens wait for the respective station to process them, N_0 is the starting location of tokens, N_{10} is the final location of tokens, and direct arrows represent transitions and component interconnections. Dotted arcs represent the particular situation where a token is re-worked in the same node without any other token being worked in the station. Dashed arcs represent the situation where tokens do not follow the standard layout of the PL, jumping to a different station or being removed from the production area (e.g., anomaly detection and removal).

General Model for a Manufacturing Production Line

To have a precise definition for the problem, with similar representation as Figure 2, the mathematical notion of a Queue Directed Graph (QDG) is introduced as an ordered tuple $(N, Q, A, B, C, K, W, G, dt_B, dt_E)$ where:

1. N is a list whose elements are called nodes with $s_N + 1$ elements and indices in the set $i \in \{0, \dots, s_N\}$;

2. Q is a list of s_Q first-in/first-out queues (FIFO queues) with no capacity limit and indices in the set $j \in \{1, \dots, s_Q\}, s_Q = s_N$;

3. A is a set with a minimum of s_N directed arcs as pairs of node-queue of the form (N_i, Q_j);

4. B is a set of s_N directed arcs as pairs of queue-node of the form (Q_j, N_i), which associates one queue to its corresponding node for all nodes in the graph except for N_0;

5. K is a list of $s_K + 1$ tokens with indices in the set $k \in \{0, \dots, s_K\}$;

6. C is a matrix of integers where $c_{i,j}$ accounts for the number of tokens that have transverse the arc (N_i, Q_j);

7. W is a list of tuples with the form $W_k = (din_w, dout_w, N_w, K_w, P_w, G_w)$ with indices in the set $w \in \{0, \dots, s_W\}$, where din_w is the timestamp when the token K_w enters the node N_w, $dout_w$ is the timestamp when token K_w leaves N_w, $din_{w+1} \geq din_w$, P_w is the duration of time that the token K_w was in N_w before leaving, and G_w is a set of words used for Root Cause Analysis (which may be empty) associated with the processing time;

8. G is a set of s_G tags (words) that are used to classify 'situations' (this set can be empty);

9. The operations on the QDG are performed in the interval $[dt_B, dt_E]$;

10. Tokens do not spend time moving on arcs (i.e. they move instantaneously);

11. The node N_0 is the starting node and node N_{s_N} is the ending node;

12. Tokens can only enter through N_0, can leave from N_{s_N}, and cannot enter the graph more than once.

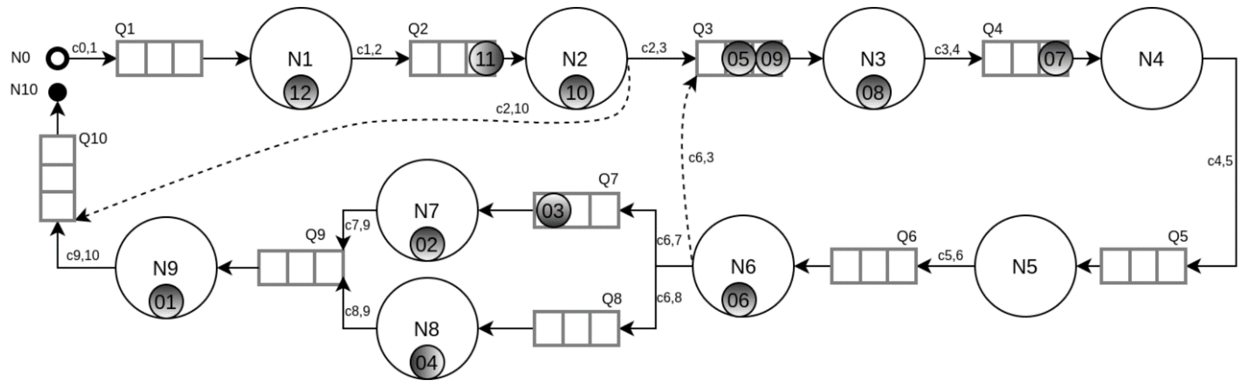
Note that, because of assumptions 3 and 4, there are no arcs from nodes to nodes or queues to queues. Moreover, because in assumption 3 the cardinality of A is not fixed, the graph transition may not be deterministic (i.e. tokens may follow different paths on exiting a node). Each token defines a path in the QDG, which indirectly is coded in the matrix C, from where one may associate to each arc a transition probability. This opens the possibility to apply Stochastic and Probabilistic Methods to QDG (e.g. Markov Chains or Bayesian Networks). Furthermore, the graph in Figure 2 is not a QDG, since it has node loops (i.e. fails assumption 12), but may be solved through operation E_1.

(E_1) As stated above, in practical situations a token (i.e. component/material/product) can be re-worked at the same node (i.e. station in the PL) without any other token entering the node. This situation is not directly compatible with a QDG, but can be treated by supposing that two consecutive entries in W as $(din_w, dout_w, N_w, K_w, P_w, G_w)$ and $(din_{w+1}, dout_{w+1}, N_w, K_w, P_{w+1}, G_{w+1})$ are replaced by the entry $(din_w, dout_{w+1}, N_w, K_w, P_{new}, G_{new})$, where $P_{new} = dout_{w+1} - din_w + P_w$ and $G_{new} = G_w \cup G_{w+1}$.

(E_2) Human operations may break predefined rules as the FIFO order of a queue. In such case and to comply to a QDG, the situation is reported and the involved tokens are completely removed from the data set for this study.

From now on, the authors assume that the above situations are treated by the operations $E_1 - E_2$, so one can have a well-defined QDG as in Figure 3.

Figure 3. Schematic representation of a Queue Directed Graph (QDG) with the associated metrics calculations.



This mathematical formulation is quite general and can be applied to many real scenarios, which will be explored in future publications.

The QDG structure, along with an integration into a technology infrastructure, allows to define metric levels. Metric levels are increasing self-contained layers where metrics (e.g. KPIs) are defined from metrics of previous levels. In the metric layer 0 sits a QDG. The authors now provide an example of two metric layers.

Metrics level 1: Considering the existence of a well-defined $QDG = (N, Q, A, B, C, K, W, dt_B, dt_E)$, one needs to define metrics related with:

- Tokens last processing locations: A list LP with the size s_K that in the position k (associated with the token of index k) holds the pair $(dout_k, N_k)$ where $dout_k$ is the last timestamp of departure of the token from the node N_k. The list starts empty;
- Node idle times: A list IT with the size s_W that holds the idle time duration (a float) for each entry W. The list starts empty.
- Node accumulated processing: A counter with a value smaller than s_K which enables to keep track of how many tokens have been processed for the desired timeframe e.g. the current shift. The counter starts as zero and may be reset according to the desired metric, e.g. daily or at shift start.
- Tokens throughput time: An attribute for each token k which measures the time elapsed in $[dt_B, dt_E]$ spend to go from the start node to the end node.

Then, for each $w \in \{0, \dots, s_W\}$, so having $W_k = (din_w, dout_w, N_w, K_w, P_w, G_w)$ and $k = index(K_w)$, the above metrics can be calculated as: (a) $IT_w = LP_{k,0} - din_w$; (b) $LP_w = (dout_w\ , N_w)$.

Metrics level 2: From the metrics obtained in the previous level, one may calculate for each element (i.e. nodes, queues, tokens) but also globally for the PL, the following metrics: (a) node idle time (from IT_w); (b) node processing time (from P_w); (c) node locking time (as the sum of idle time plus processing time); (d) node starving time (as the difference between the total study duration and the total locking time); (e) node throughput (as the number of token leaving a node per time unit); (f) queue permanence (as the number of token in the queue per time unit), etc..

The authors would like to remind the reader to the fact that, although this mathematical approach is quite relevant for the purpose of this study, it is still general enough to accommodate many other problems, e.g. bottleneck detection and prediction or predictive quality control. This gains particular relevance when one considers the usefulness of level 2 metrics and higher.

Workstation Processing Times Data Generator

Although the proposed approach will be tested against data from a real manufacturing PL, to be able to proceed with several experiments regarding parameter stability and prediction robustness, fundamental to benchmark the chosen problem and also the machine learning approach, it is required to have a mechanism to generate mathematical valid data from the real data. In the literature, the authors were able to find works regarding simulation and data generation of data for a PL as Centomo (2018). However, the mentioned approaches made too many simplifications or are not easily adapted to the problem under study. Additionally, the purpose here is to find a mathematical model as accurate as possible to this situation.

In this work, in order to generate data under valid assumptions, a QDG model is used where artificial data is obtained by simulation by using two generators to construct the values of W_k (see assumption 7 on the definition of a QDG): (a) a node-to-node path generator for each token; and (b) a processing times generator for each node. The path generator is simply obtained from real data by constructing a count transition matrix node-to-node, from which we obtain the probability of a token leaving a node i and moving into a node j. The processing times generator is not so straight forward and needs to be adapted to the real data, as much as possible.

A common approximation for the processing times (PT) distribution is to use a normal distribution. However, such is quite imprecise, since processing time histograms of real data show that the data is not symmetric, does not have negative values or even values below some positive threshold, and outliers tend to follow a different distribution. Such implies that a more realistic distribution function for the generator will be a composite probability distribution, see Oosterbaan (2019).

As expected, measures between probability distributions play a significant role in the quest to find a good generator. There are many variants such as the Hellinger coefficient, Jeffreys distance, Chernoff coefficient, directed divergence, and Kullback–Leibler divergence (relative entropy). Using the relative entropy, the probability density function (p.d.f.) of processing times of real data and the p.d.f. associated to a processing times generator may be compared to measure how near they are to each other. Then, by experts' intuition and tentative and error, several processing times generators may be created and tested to find the best candidate. The generator will depend on parameters that take into account the variability and specificity of each workstation, which can be found by traditional grid fitting methods on real data. With the above approach, the generator proposed in this work is defined as follows:

Let $U(a,b)$ and $LN(\mu,\sigma)$ denote (respectively) a uniform distribution defined in $[a,b] \subset \mathbb{R}$ and a log-normal distribution with location parameter μ and shape parameter σ. Define the constants $0 < a_1 < b_1$, $a_2 < b_2, 0 < a_3 < b_3, 0 \leq a_4 < b_4 < 1$, the maximum percentage of outliers $r \in [0,1[$, and for each processing location $i \in \mathbb{N}$:

- $w_i \in U(a_1, b_1)$ as the lower production barrier;
- $\zeta_i \in U(w_i + a_1, w_i + b_1)$ as the upper outlier barrier;
- $\mu_i \in U(w_i + a_2, w_i + b_2)$ as the location parameter (e.g. mean);
- $\sigma_i \in U(a_3, b_3)$ as the shape parameter (e.g. standard deviation);
- $\gamma_i \in U(a_4, b_4)$ as the outlier selection parameter;
- the generator function for the processing times as

$$P_i = \begin{cases} U(w_i, \zeta_i) & , if\ r < \gamma_i, \\ w_i + LN(\mu_i, \sigma_i) & , if\ r \geq \gamma_i. \end{cases}$$

In this definition, P_i behaves as a random variable with bounded support that generates the processing times at the node i. The first branch generates abnormal processing times and the second generates the normal ones. The distribution has support above w_i because it is not realistic to assume that a machine or an operator can shrink the processing time to zero, without a lower limit, just by adding more effort without scenario changes (e.g. increasing hardware or human resources). As expected, the above parameters can be precisely found for a given data set but also allow to model different data behaviors on each processing location.

As is predicted by the famous Central Limit theorem, the distribution of the sum of (a big enough) number of processing times at different locations will approach a Normal distribution, which is reinforced by the fact that some products may leave the PL earlier (e.g. defect detection at a station), thus contributing to the symmetry of the distribution of the throughput time as a sum of the distributions of the processing times at each workstation.

THROUGHPUT PREDICTION ON SIMULATED DATA

The main goal of this section is to describe the various tests that were conducted in order to find out whether it would be reasonable for the proposed framework to predict in real time the efficiency of a given token. In practice, this means that the authors have started with a feature of the graph such as the throughput time (e.g. the time it takes for a token to navigate the graph from start to end). Based on a given percentile one calculates a label based on it: 0 (not efficient) or 1 (efficient). What will be assessed is whether one can accurately predict this label by using incomplete information about the feature at hand. For this use case, the first nodes and transitions durations will be used in order to predict the label's outcome. If the authors succeed, then one can predict a token's performance using just the data from the first nodes. Since this is a graph, its length is completely irrelevant since this applies to all use cases. What might prove meaningful is the ratio between the nodes used for prediction and the number of nodes in the graph.

Parameter Stability and Predictive Robustness

To have a valid industrial solution, it is of major importance to measure its stability to a change in the conditions. This will have a consequent impact in the robustness of the algorithmic solutions used. The following experiments have been conducted using simulated data as described in the *Workstation Processing Times Data Generator* subsection, in order to have a data set with the intended characteristics. Note that the data sets generated with the distribution proposed are similar (in a precise mathematical sense) to a data set with data of a real production line. A graph with a typology of 9 nodes has been used.

Experiment A – Class of machine learning algorithms: The first step to have a controlled study about parameter stability and predictive robustness, allowing the use of hyperparameter and pipeline optimization, is to define a class of techniques to optimize and test. As described in the introduction, the Machine Learning (ML) problem to be solved may be reduced to a binary classification.
There are many techniques in the literature that may be applied to binary classification with proved success such as: (a) Naïve Bayes, (b) Logistic regression and variants; (c) k-Nearest neighbor with Euclidean or Mahalanobis distances; (c) Support vector machines and variants with kernel functions; (d) Variants of

decision trees as bagging decision trees, boosted decision trees, or random forests; (e) Gaussian mix models; and (f) Voting classification.

After testing each of the above techniques in the data sets of this work with sparse grid hyperparameter optimization (whose descriptions are omitted), the conclusion is that all techniques perform relatively well. However, the variants of decision trees are the best ones when compared to the standard metrics obtained from the confusion matrix. For such reasons, from now on, the experiments use techniques such as these and were implemented in standard Python libraries such as Scikit-learn and TPOT, for automated machine learning (autoML).

Although the percentile threshold predefined will characterize if the classes are balanced or not (i.e. being near 75%, mean that it exists 75% of zeros and 25% of ones), the number of false positives and false negatives obtained were quite low, so the focus will not be on F1 but rather on the accuracy metric.

Experiment B – Number of nodes and percentile influence: This experiment intends to understand if there is a dependence between the accuracy metric results obtained by a Random Forest Classifier (RFC) and the changing number of features (i.e., number of nodes used for prediction). The label is obtained from the percentile of the total time that tokens are in the graph (i.e., going to all the nodes and queues of their complete paths).

Using the standard Scaler in conjunction with RFC with 100 estimators, and without any further fine tuning to the pipeline, the accuracy results described in Table 1 were achieved. Two thirds of the data were used for training whilst the remaining were used for testing.

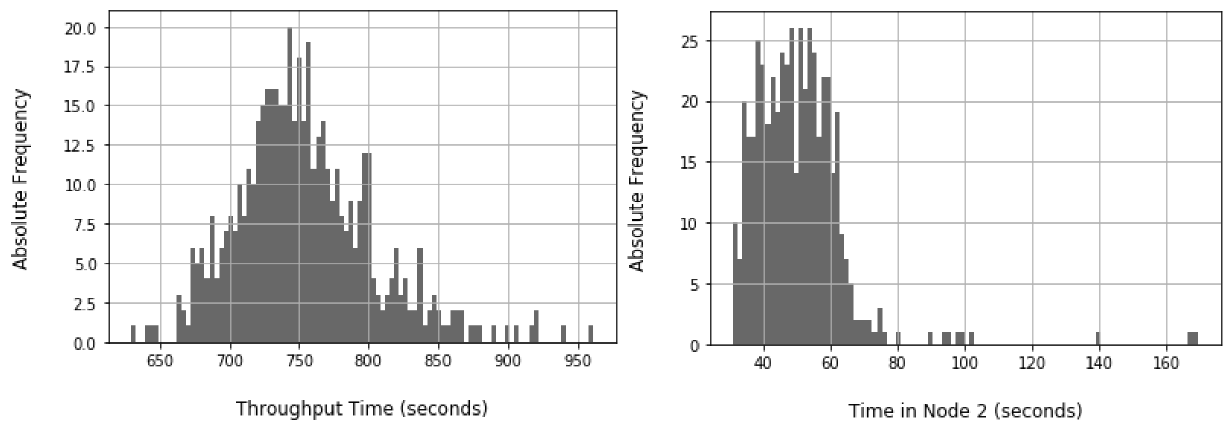
Here, the training algorithm was run for two different percentiles to assess the impact of the percentile in the overall outcome. Additionally, the authors have also ran a single data set and a data set with enriched features. Single features are the durations of nodes and transitions (time spent in the queue), while enriched features include some of these durations summed together. Since these aid the algorithm to find correlations, one can see a significant increase in the accuracy when the enrichment is added. The only exception seems to have occurred for the 2 out of 9 nodes experiment with percentile 80%. This deviation may be due to inherent characteristics of the data set.

Table 1. Results for experiment B, where the accuracy of the pipeline was tested with different levels of information and against different percentiles

	Percentile = 60%		**Percentile = 80%**	
	Single Features	**Single + Enriched Features**	**Single Features**	**Single + Enriched Features**
2 out of 9 nodes	58.18%	65.45%	79.39%	78.18%
3 out of 9 nodes	66.66%	70.30%	78.78%	81.21%

One can here observe already very satisfying levels of prediction which indicate that the data set is quite stable. This can be confirmed by analyzing the histogram of the throughput time from which the percentile is calculated, see Figure 4 (left). One verifies that the percentile introduces some perturbation in the accuracy of the classifier thus making it perform better for higher percentiles.

Figure 4 (left) and Figure 4 (right). On the left, histogram from the metric from which the percentile is calculated. On the right, histogram of the processing time in the second node of the graph. The number of bins was adjusted to provide enough granularity.



Experiment C – Outlier influence: In this experiment, the outlier variation was set to an uniform distribution with parameters 0 and 0.5 i.e. there was a significant reduction of the outlier variation in comparison to Experiment B.

Table 2. Results for Experiment C where the accuracy of the pipeline is tested against a more reduced level of outliers.

	Percentile = 80%	
	Single Features	**Single + Enriched Features**
2 out of 9 nodes	78.79% (-0.6%)	79.39% (+1.21%)
3 out of 9 nodes	81.81% (+3.03%)	83.03% (+1.82%)

The 60% percentile was dropped as this is not a sufficiently realistic percentile level for real applications. Compared with Experiment B (see Table 1), one can see that in the first row of Table 2, there is not much improvement of the classifier accuracy. Most probably, this is due to very little data being used, which translates into higher deviations.

In the second row of Table 2, there was a slight increase in accuracy for both single and enriched features compared to Experiment B. The underlying reason is that the enriched features tend to soften the variability that is introduced by the outliers.

Experiment D – Data set variability: In the previous experiments the ML pipeline was tested in one data set only. In this experiment, 5 different simulated data sets are used, in order to study the robustness of the classifier. Outlier variability is the same as in the previous experiment.

Table 3. Results for experiment D where the accuracy of the pipeline is compared among different data sets

		Percentile = 80%	
		Single Features	**Single + Enriched Features**
Data set 0	2 out of 9 nodes	73.94%	73.94%
	3 out of 9 nodes	76.36%	79.40%
Data set 1	2 out of 9 nodes	76.36%	78.79%
	3 out of 9 nodes	84.24%	86.67%
Data set 2	2 out of 9 nodes	78.18%	78.79%
	3 out of 9 nodes	81.82%	81.82%
Data set 3	2 out of 9 nodes	78.18%	79.39%
	3 out of 9 nodes	80.61%	81.21%
Data set 4	2 out of 9 nodes	77.58%	79.40%
	3 out of 9 nodes	82.42%	83.03%

Table 4. Statistics for the results in Table 3.

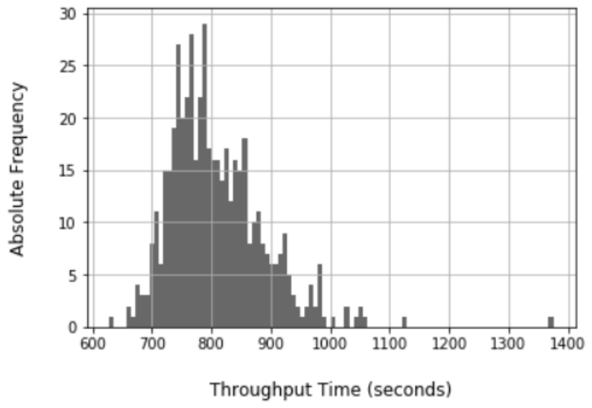
		Minimum	**Mean**	**Maximum**	**Δ (Max-Min)**
2 out of 9 nodes	Single Features	73.94%	76.84%	78.18%	4.24%
	Single + Enriched Features	73.94%	78.06%	79.40%	5.46%
3 out of 9 nodes	Single Features	76.36%	81.09%	84.24%	7.88%
	Single + Enriched Features	79.40%	82.42%	86.67%	7.27%

Overall, the classifier robustness is within reasonable limits of variability. The interval between maximum and minimum accuracy increases as more nodes are used (see Table 4). This makes sense since the number of features is also being increased and thus variability increases.

Again, the smoothing effect provided by the enrichment is an advantage for classifier's performance as one can assess by comparing single versus singles plus enriched features. This holds even for this larger sample as expected.

Experiment E – Local and global perturbations (location parameter): Up to now, the simulator was using a range of parameters for the log normal distribution for the duration in each transition and node. In this experiment a deviation will be introduced in Node 2. The goal is to test how this deviation affects the overall classifier performance, which is tied to the end metric, the Throughput Time. Here, the parameter μ of the log normal (i.e. the location parameter) is shifted by an additional 2 units. These units are added to a parameter which is generated from an uniform distribution.

Figure 5 (left) and Figure 5 (right). On the left, the histogram from the metric from which the percentile is calculated. On the right, histogram of the duration in Node 2 where the introduced deviation is visible.



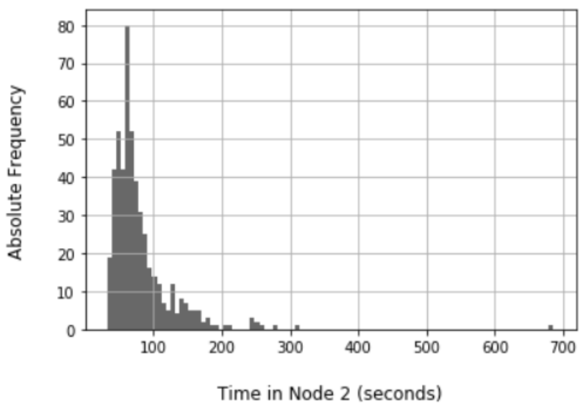


Table 5. Results for experiment E where the accuracy of the pipeline is compared among different data sets with the introduced perturbation.

		Percentile = 80%	
		Single Features	**Single + Enriched Features**
Data set 0	2 out of 9 nodes	79.40%	81.82%
	3 out of 9 nodes	83.03%	82.42%
Data set 1	2 out of 9 nodes	80.61%	81.82%
	3 out of 9 nodes	81.21%	81.21%
Data set 2	2 out of 9 nodes	81.82%	78.18%
	3 out of 9 nodes	86.06%	83.64%
Data set 3	2 out of 9 nodes	80.61%	81.21%
	3 out of 9 nodes	82.42%	81.21%
Data set 4	2 out of 9 nodes	76.97%	78.79%
	3 out of 9 nodes	80.00%	80.00%

Table 6. Statistics for the results in Table 5.

		Minimum	**Mean**	**Maximum**	**Δ (Max-Min)**
2 out of 9 nodes	Single Features	76.97%	79.88%	81.82%	4.85% (+0.61%)
	Single + Enriched Features	78.19%	80.36%	81.82%	8.63% (+3.17%)
3 out of 9 nodes	Single Features	80.00%	82.55%	86.06%	6.06% (-1.82%)
	Single + Enriched Features	80.00%	81.70%	83.64%	3.64% (-3.63%)

In overall terms, the classifier's performance is equivalent to that of Experiment D. Regarding classifier stability, there is a trend for an increased amplitude in the 2 out of 9 nodes scenario. Yet on the 3 out of 9 nodes, the amplitude shows a decreasing trend which translates in better stability even when one tends to the fact that a significant deviation was introduced.

***Experiment F – Local and global perturbations (shape parameter)**:* This experiment follows the logic of the previous experiment with the difference that the parameter now shifting is the σ of the Log-Normal (i.e. the shape parameter). This shift is also 2 units. This perturbation introduces a much deeper disturbance to the distribution as Figure 6 shows.

Figure 6 (left) and Figure 6 (right). On the left, histogram from the metric from which the percentile is calculated. The effect of the deviation is quite drastic. On the right, the histogram of the duration in Node 2 where the introduced deviation is also clearly visible.

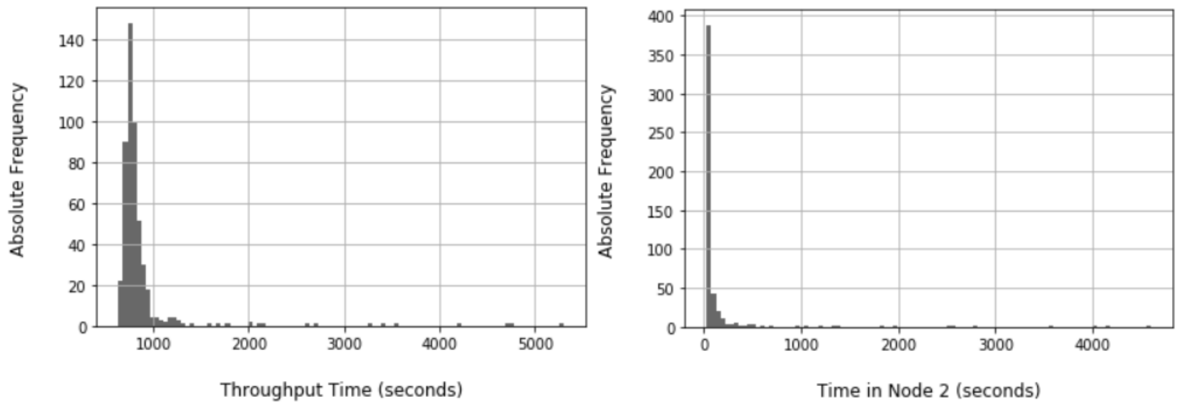


Table 7. Results for experiment F, where accuracy of the pipeline between different data sets was compared.

		Percentile = 80%	
		Single Features	**Single + Enriched Features**
Data set 0	2 out of 9 nodes	86.06%	84.24%
	3 out of 9 nodes	86.06%	85.46%
Data set 1	2 out of 9 nodes	89.09%	90.30%
	3 out of 9 nodes	88.49%	89.09%
Data set 2	2 out of 9 nodes	85.46%	84.24%
	3 out of 9 nodes	86.06%	87.88%
Data set 3	2 out of 9 nodes	87.88%	88.48%
	3 out of 9 nodes	90.30%	90.30%
Data set 4	2 out of 9 nodes	85.45%	87.27%
	3 out of 9 nodes	89.09%	88.49%

Table 8. Statistics for the results in Table 7.

		Minimum	**Mean**	**Maximum**	**Δ (Max-Min)**
2 out of 9 nodes	Single Features	85.45%	86.79%	89.09%	3.64% (+0.6%)
	Single + Enriched Features	84.24%	86.91%	90.30%	6.06% (-0.6%)
3 out of 9 nodes	Single Features	86.06%	88.00%	90.30%	4.24% (+3.64%)
	Single + Enriched Features	85.46%	88.24%	90.30%	4.84% (+2.43%)

The perturbation introduced has improved greatly the classifier's performance. Classifier robustness has also improved in global terms.

Behavior Prediction through Automated Machine Learning

After a good understanding of the classifier's behavior gained in the previous experiments, now the authors apply AutoML to see how to further optimize these pipelines. This is a task where TPOT (Tree-based

Pipeline Optimization Tool) can be of great assistance. TPOT is a Python library which tests different ML pipelines against a data set, see Randal (2016). Not only that, it also fine-tunes the hyperparameters of the prediction algorithms. TPOT is based on genetic programming and the user can configure TPOT to run based on a set of inputs. The reason behind the use of this strategy had to do with the very good performance that the Random Forest Classifier already exhibited in the previous experiments. Since TPOT is also tree based, this is a logic next step.

TPOT's Classifier uses randomness to find the best pipeline combination. Due to this, the same data set may be fed to TPOT using the same input parameters and produce different results. One way of making its results replicable, is to introduce the RANDOM_STATE variable in its input so that the same code may be ran twice and output the same pipeline as a result.

Although TPOT is immensely useful, it takes some tries to get it to run and finish the pipelines optimization. One of the causes for this behavior is that it requires a lot of computational resources and without the right configuration parameters it may crash. Another important aspect is that TPOT relies on other libraries such as Sci-Kit Learn, XGBoost or PyTorch which makes it particularly complex to set up. On the other hand, this is an advantage as these are popular ML libraries and their ML algorithms are open source and well-known within the scientific community.

TPOT was applied for the same 5 data sets that were used in Experiment D. The results are presented in Table 9. This run was executed with GENERATIONS = 5, POPULATION_SIZE=10, N_JOBS = -1 and N_SPLITS = 10.

Table 9. Results obtained using TPOT for 5 different data sets with percentile set at 75%.

	Accuracy Test Set	**Accuracy Training Set**	**Pipeline**
Data Set 0	100%	99.87%	GradientBoostingClassifier(learning_rate=0.5, max_depth=8, max_features=0.25, min_samples_leaf=19, min_samples_split=19, n_estimators=100, subsample=0.85)
Data Set 1	100%	99.93%	XGBClassifier(learning_rate=0.001, max_depth=8, min_child_weight=6, n_estimators=100, n_jobs=1, subsample=0.55, verbosity=0)
Data Set 2	100%	99.80%	XGBClassifier(learning_rate=0.01, max_depth=2, min_child_weight=2, n_estimators=100, n_jobs=1, subsample=0.3, verbosity=0)
Data Set 3	99.8%	100%	DecisionTreeClassifier(criterion="entropy", max_depth=4, min_samples_leaf=6, min_samples_split=13)
Data Set 4	100%	99.73%	StackingEstimator(estimator=GradientBoostingClassifier(learning_rate=0.01, max_depth=7, max_features=0.2, min_samples_leaf=8, min_samples_split=5, n_estimators=100, subsample=0.8)), GaussianNB())

At a glance, it may look like there is some overfitting occurring. However, remember how in Experiment D already very good accuracies were achieved without the need to make any fine tuning to the hyperparameters of the Random Forest classifier. If the examples of Data Set 1 and Data Set 2 are to be taken into consideration, since they are optimized by the same pipeline, there is already an accuracy of 100%. This also describes the relative stability of the process itself that is being modeled.

In most scenarios, the pipeline is composed of an unique classifier. The exception is Data Set 4 where a stacking estimator is used. A stacking estimator uses different classifiers (in this scenario only two) and then uses the best outputs to maximize its accuracy. This is another advantage of using an AutoML tool such as TPOT. These combinations are done automatically. Otherwise, they might not occur to the Data Scientist or the number of combinations may be too cumbersome to test to a full extent.

Even though the authors have achieved some satisfying results in this section, there is now the need to test this methodology with real data in order to verify whether the classifier's characteristics hold.

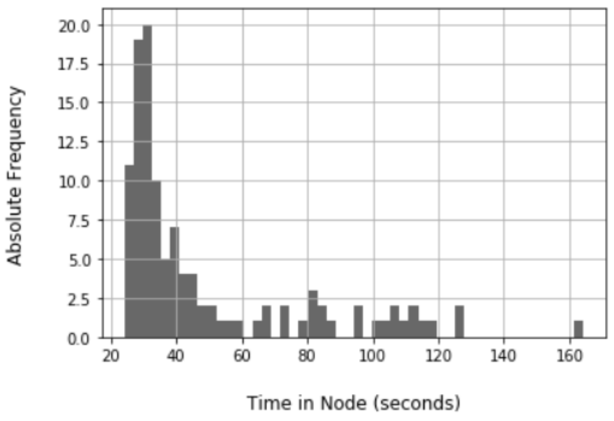
THROUGHPUT PREDICTION ON MANUFACTURING PRODUCTION LINE

With the goal of testing the hypothesis at hand, in this work a data set which originates from an actual production line located in the Bosch Thermotechnology's Aveiro plant in Portugal was used. The data set was obtained from the Manufacturing Execution System (Bosch's Nexeed MES) and converted in order to match the structure of the general mathematical model. This line is comprised of a total of 18 workstations and it operates on 2 to 3 daily shifts. The yearly production for this line is around 50.000 parts. In Table 10 and Figure 7, the authors characterize one of these stations through some data statistics.

Table 10. Data descriptive statistics for one of the stations of this line for the processing time feature for the data set of one shift.

Statistics for Processing Time in the station	
Minimum	24.21 s
Average	48.49 s
Median	34.29 s
Maximum	164.25 s
Standard Deviation	29.40 s
Q1	29.61 s
Q2	34.23 s
Q3	50.67 s
Count	113

Figure 7. Histogram of the data characterized in Table 10.



Dimension Reduction and Visualization

Dimension reduction is a standard approach to identify relevant features and remove others that do not give a significant contribution to the classifier’s task. Although, there are many techniques for dimension reduction, the most popular technique is principal component analysis and its variants, see Liu (2007) and references within. Considering the mathematical characteristics of the data sets and the need to visualize the data topology (as reduction from the number of features to dimension 2), the t-distributed stochastic neighbor embedding was also implemented, see Maatens (2008).

Here, two data sets are considered: (A) having values for 8 elements; (B) having values for 4 elements; with the same accumulate production line transverse time with a total of 18 nodes.

PCA - Principal Component Analysis: PCA was introduced by Karl Pearson back in 1901 and there are many variants in the literature, e.g., robust PCA, kernel PCA, or localized PCA, see Ma (2019). Here, standard PCA is used. However, since PCA may be sensitive to feature scale and outliers, different scaling methods were tested. These feature transformations are comprehensively explained in SciKit-Learn’s documentation, see Sci-Kit Learn (2020). It was observed that although the scaler choice is important in many scenarios, this was not the case here. In Figure 8, one can observe the results of such implementation for both data sets.

Figure 8 (left) and Figure 8 (right). Cumulative explained variance plotted (in the y-axis) and the number of features (in the x-axis) after PCA application using the Standard Scaler for the A and B data sets, respectively.

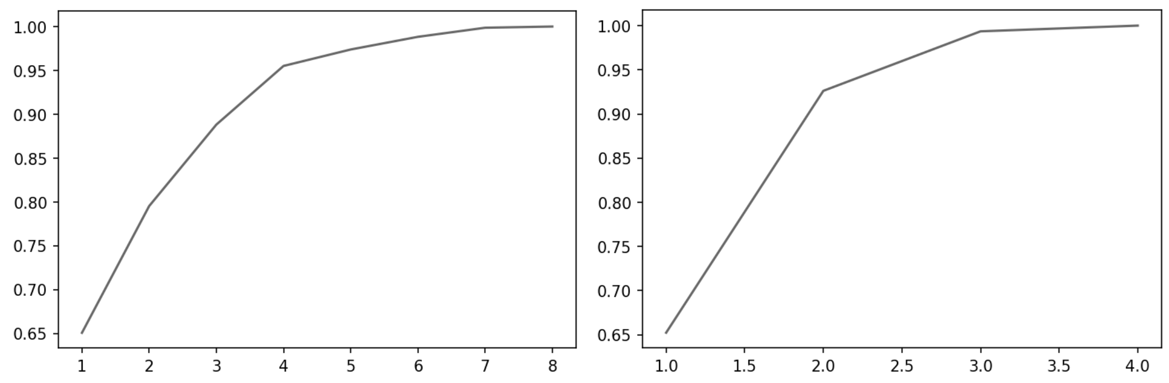
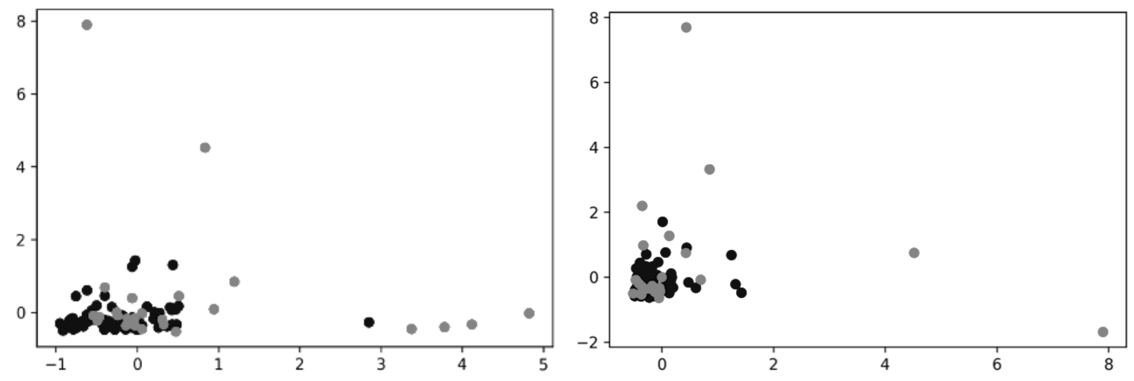


Figure 8 (left) shows that a relatively small number of elements is enough to characterize 90% of the data. Such conclusion is useful for applications in real situations.

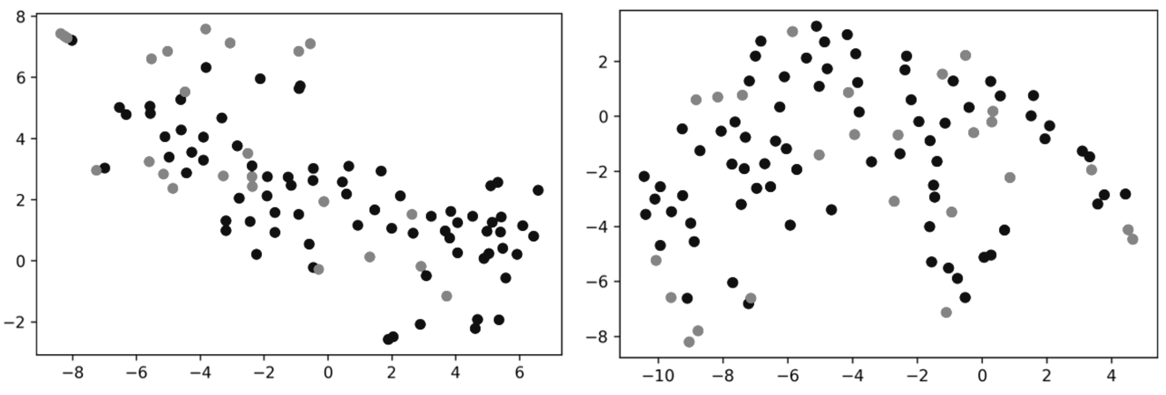
Figure 9 (left, situation A) and Figure 9 (right, situation B). Darker values are below percentile 75%; lighter values are equal or above percentile 75%; the x-axis is related with the first principal component and the y-axis is related with the second principal component.



Plots in Figures 9 are obtained by using PCA (with whiten and no renormalization for dimension reduction) for both situations. Although, a smaller number of elements characterize the data, the reduction to dimension 2 shows that the different classes are highly overlapped. Something expected since the elements nodes that are not in the data may be a source of increased processing time.

t-SNE – t-distributed Stochastic Neighbor Embedding: t-SNE is a very effective unsupervised nonlinear method for data visualization. While t-SNE preserves only small pairwise distances or local similarities, PCA is concerned with preserving large pairwise distances to maximize variance. Laurens illustrates the PCA and t-SNE approach pretty well using a Swiss Roll; if data has a spiral format distribution, PCA will connect points that are close in Euclidean distance sense but such points may be quite far when one follows the spiral structure.

Figure 10 (left, situation A) and Figure 10 (right, situation B). Darker values below percentile 75%; lighter values equal or above percentile 75%; the x-axis is related with the x-coordinate of the net elements and the y-axis is related with the y-coordinate.



From analyzing the results depicted in Figure 10, and since the unknown operation to predict has a linear structure, it is expected that data sets with a smaller number of elements will not lose too much information and by finding another well-suited ML pipeline, the classifier metrics will be similar to data sets with a higher number of elements information. Furthermore, there is no clear clustering structure for the label data (on the figures of both techniques) so linear ML techniques are not expected to perform well without applying an adapted lifting function to take advantage of the kernel trick. In other words, the classification problem is not trivial by itself.

Prediction Results

In Table 11 the results of predicting the correct label using from two to five sequential features are presented. This run was executed with GENERATIONS = 5, POPULATION_SIZE=10, N_JOBS = -1, N_SPLITS = 10 and N_REPEATS = 3, the same inputs as previously. In practice, this represents how accurately one can predict the throughput time of a product based on the outcome at the early stages of the graph.

Table 11. Results with the line's data set where TPOT optimized different pipelines for each use case with a 75% percentile.

	Accuracy	Pipeline
5 of 18 nodes	97.50%	StackingEstimator(estimator=MLPClassifier(alpha=0.01,learning_rate_init=1.0)), StackingEstimator(estimator=GaussianNB()), DecisionTreeClassifier(criterion="entropy", max_depth=9, min_samples_leaf=3, min_samples_split=11)
4 of 18 nodes	96.59%	StackingEstimator(estimator=DecisionTreeClassifier(criterion="entropy", max_depth=3, min_samples_leaf=20, min_samples_split=17)), BernoulliNB(alpha=10.0, fit_prior=True)
3 of 18 nodes	95.76%	DecisionTreeClassifier(criterion="entropy", max_depth=6, min_samples_leaf=10, min_samples_split=5)
2 of 18 nodes	86.36%	RandomForestClassifier(bootstrap=False, criterion="entropy", max_features=0.05, min_samples_leaf=1, min_samples_split=16, n_estimators=100)

One conclusion seems to be that the simulator uses a probabilistic distribution which describes the physical phenomenon in a very good way. As it can be seen from Tables 10 and 11, the accuracy did not decrease significantly. This is a positive remark since drastic drops in accuracy are usually a common symptom of a rough theoretical approximation model to generate the synthetic data.

It might be surprising that the accuracy is so high in these scenarios. Since a label is being predicted which is tied with the throughput time of the graph, then the correlation that the ML algorithm is trying to find is actually a linear combination of the time spend in visible and hidden nodes/queues. Since TPOT explores a number of combinations between pipelines and the features provided, it is also not surprising that the results have such a high accuracy. However, a high accuracy may be the result of overfitting the training data with patterns specific to a particular (train, test) pair set.

From the start, the authors had always in mind that some challenges would probably arise, and this was indeed what was observed. Since the domain is organizational, there are several issues that need to be addressed.

Challenges for the Deployment of the Solution

With the increasing availability of data in many organizational departments, the urgency to put this data to good use is increasing at a steady pace. However, sometimes the challenges seem unsurmountable. From organizational to technical difficulties, there is still a long road ahead for most organizations to be able to get the most leverage out of their data and solutions as the one described with simulated data.

In this subsection, one will explore the types of challenges being faced. These challenges have been split into three categories: (i) organizational, (ii) development, and (iii) technical and data pipeline.

Organizational challenges

Issues, controversies, and problems: As mentioned in Hartley (2019), digitalization is a process, and a long one indeed. From the start, it is important to convey to everyone in the organization the benefits and added value that these technologies bring. The human factor is generally one of the first barriers that digitalization faces.

People are at the center stage and they are the main drivers of this line of work. Data related techniques can be used in more traditional fields such as prediction or optimization, but they are starting to be increasingly used in other fields such as innovation, see Haefner (2021). As a result, most organizations are already including projects in their roadmaps which include techniques related with Machine Learning. However, putting these ideas into practice is very challenging in itself.

One of the first limitations encountered is related with talent management and enabling. It is often the case that very few people in the organization have the right know-how to conduct a long-term data strategy. With so many new technologies being released in very short timeframes, it is very easy to lose track or even know where to begin learning. This may occur even for highly trained and specialized engineers/IT personnel.

Moreover, even if the organization does have this kind of talent within its ranks, there is often a huge gap between collaborators which are experts in the organization's processes (being them from the manufacturing, logistics or financial areas) and the ones more focused on the technical side (infrastructure, mathematical techniques, etc.). This makes communication very hard and frustrating and causes many disturbances to the whole process.

Recommendations: The authors have explored previously how building know-how in technical and processual topics is a hindrance in today's organizations. There are different strategies to work around this challenge. One might be hiring someone with the right background in order to support the company to establish its own data strategy, not only in terms of the infrastructure but also for the overall, log-term data strategy. This could be a suitable strategy for Small and Medium Enterprises (SMEs). However, most often organizations will need a complete team to support this need. This sort of work involves almost the entire organization and requires extensive support from the higher management level in order to be able to work effectively due to the required changes it encompasses.

Another alternative, which is precisely the case at hand here, is the establishment of partnerships between local universities and organizations. This can translate into a tremendous advantage since both parties have a lot to contribute and benefit from working hand in hand. This combination of cutting-edge techniques and technologies with the field experience of collaborators of the organization is invaluable and the authors believe that this partnership will yield very significant results over time.

There was also the mention to the fact that there is often a huge gap between engineers and process experts. One recommendation to overcome such difficulties is to involve a third party which acts as a mediator between them. As data driven projects become more disseminated among organizations this need might tend to fade over time. At this point, it becomes clear that the organization is shifting towards more data awareness.

Development challenges

Issues, controversies, and problems: Besides the challenges already mentioned which were more related with data acquisition and standardization, other important aspects also arise when it comes to developing and validating a ML framework which needs to process a great amount of information in real time. At the end of the first stage of development, the issue of verifying the correctness of the calculated metrics started to be a real issue. At first, the authors were using simulated data which was fed in real time to the system. Although this method does mimic the actual system in a comprehensive way, it was quite difficult to assess from the event's timestamps alone what was happening. Using real time data makes this process very time consuming and exceptionally prone to error. It is manageable with only one token in the system, but this is not a realistic approach to test and validate the framework's correctness.

Recommendations: In order to overcome this difficulty, a simulator was created to generate comma separated files which simulate the events that are expected to occur in the graph. This allows the authors to automatically associate to a certain token, the various timestamps and their calculated metrics. Another feature of this simulator is that it generates events following different probabilistic distributions whose parameters can be fully configured.

This simulator's use is actually two-fold: besides calculating the associated metrics, it also allows one to use the result to benchmark different ML pipelines against the data and to verify the accuracy of the prediction that can be achieved. When generating simulated data, multiple files are generated so that the ML pipelines can be tested against different data sets and thus assess the stability of the accuracy of the classifier.

Besides data validation, it is also worth mentioning that data privacy is a real concern and something which must be preserved at all times. For instance, in this scenario where a product's performance in a line is to be assessed, the identity of the operators working on the line is never to be disclosed and is completely unknown to the framework. An excellent reference on how ML models can have a hazardous influence in the day-to-day life of citizens and how to prevent these nefarious effects can be found in Kathy O'Neil's book, *Weapons of Math Destruction* (O'Neil, 2016).

Technical and data pipeline

Issues, controversies, and problems: There are no data driven projects without data. This is a fundamental truth, yet it is very easy to overlook whether a digital foundation exists in an organization. At its core, there are generally different Information Systems (ISs) which support all the business needs of these entities. Most often, these are Enterprise Resource Planning systems (ERPs) which combine a myriad of functionalities that all organizations need and rely on e.g. Human Resources or Financial processes. When one tends to look at the Manufacturing area, MESs are often a reality, but this might not be the case everywhere. This very diversified ecosystem means that having a data strategy is indeed mandatory. Otherwise, extracting actual information from such an infrastructure will be very time consuming and unproductive. This can be due to limited connectivity between the ISs and equipment yet one of the most common sources is the heterogeneity of the data sources themselves, which do not combine when one tries to fit them together.

Additionally, organizations usually face the ordeal of having to deal with legacy systems which have been in place for several years without a proper upgrading roadmap. Sometimes it may even be the case that the inner works of these systems are only known by a very limited amount of people which aggravates the problem. Most often, these are critical systems and the enterprise cannot fully function without them. This very entrenched dependency makes any effort of migration seem overwhelming due to the high risks it imposes. The same holds true for legacy equipment such as Programmable Logic Controllers (PLCs) and Industrial Personal Computers (IPCs) which have been installed many years ago and although these still supply critical functions to the manufacturing process, their connectivity is often quite limited thus not allowing these equipment to connect with the ISs in place. Most often, these devices need to be replaced by more recent ones with better functionalities, yet this transition is a very slow process. This happens not only due to technical challenges but also because of the disturbances its replacement may have in the productive process.

As one can picture and since the foundation is quite diverse, both in structure and even technologically wise, interfacing these systems with a ML framework is quite complex. Most organizations rely on flat file exports to fulfil these needs and having more advance features such as real time processing is not planned for the foreseeable future. Besides this, there is hardly ever an infrastructure in place which can meet the organization's data needs. This is often one of the first steps towards running data driven projects.

Finally, the fact that technologies are ever changing means that they become obsolete in a very short period of time. This is an added effort for organizations: besides needing to add more focus on maintaining their software up and running, technical skills are quickly deemed outdated as well. It was with this scenario in mind that a new approach was designed, one that could help organizations overcome some of these issues.

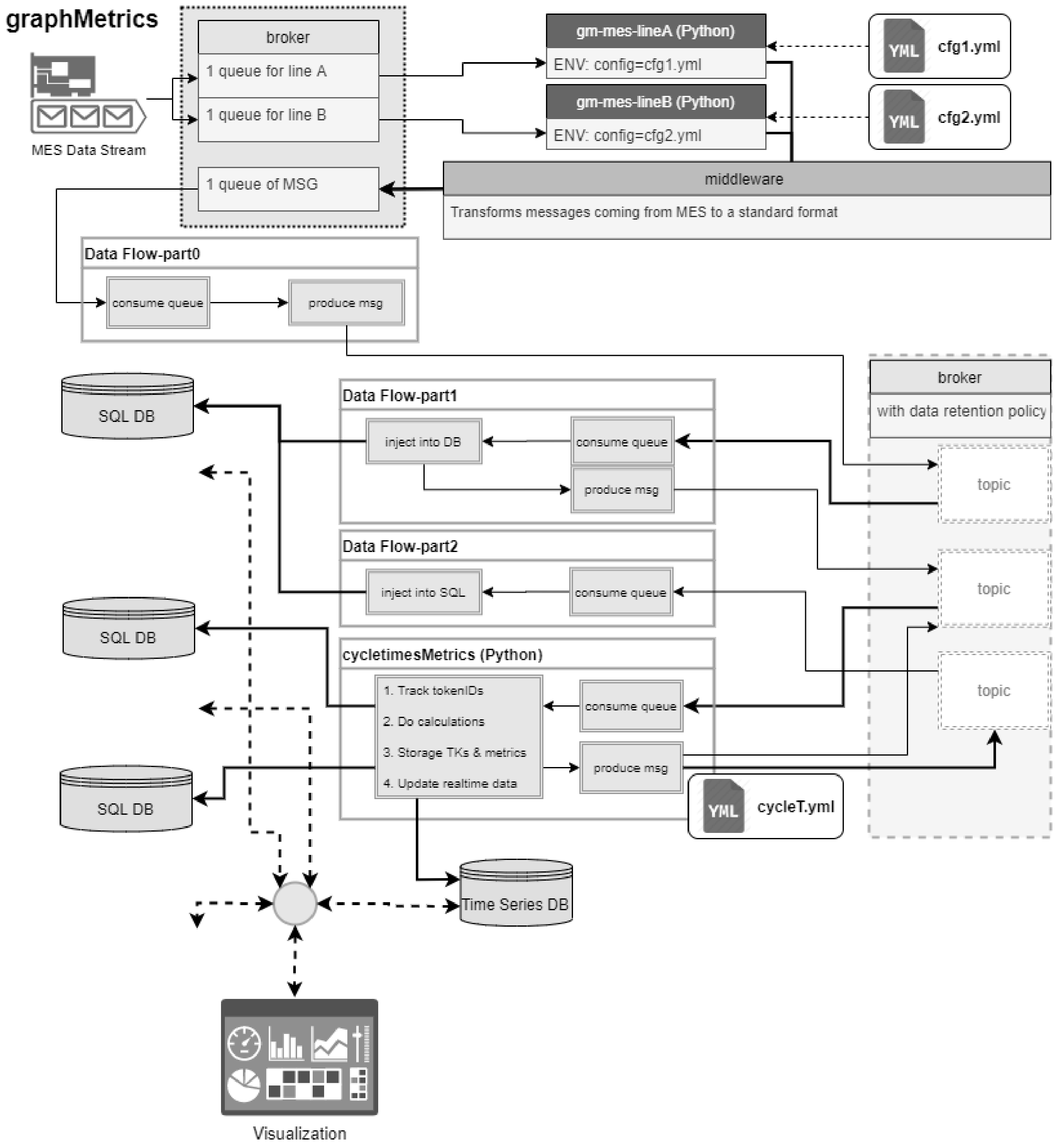
To sum up, the most prominent issues in this area are the lack of standardization in communication protocols and data structures, aged or missing technological infrastructure and poor interfaces between ISs.

Recommendations: The proposition is that a good data pipeline is not enough by itself, but together with a well characterized mathematical representation of the system relations and data, can solve multiple data-driven problems across an organization. The main idea behind this framework is that by being highly concept oriented together with real time data, a generic tool is achieved which is useful for a myriad of different contexts and situations.

In order to fulfil the challenges previously mentioned, the authors have designed a framework which has the following characteristics:

- Asynchronous Operation: Metrics for the graph are calculated asynchronously from the information currently reaching the system.
- High Configurability: The module relies on YAML files to allow the user to completely configure the process to be modeled by the abstract entity of the graph. For instance, if a different event defines the start of the graph, this can be configured.
- Easy Deployment: The architecture is implemented through the use of containers and the deployment process is carried out through a single text file. This makes deployment to production systems easy and consistent. This architecture enables a high degree of abstraction in relation to the Operation System (OS).
- Fault Tolerance: This was another important aspect taken into consideration and this is the reason why the usage of brokers and flow components is so extensive. In case of failure, no data gets lost. As soon as the service is available again, it is possible to resume operation starting from the moment of failure.
- Flexibility: This architecture was conceived with flexibility in mind so that not only systems such as MES can feed it but other data sources as well such as Internet of Things (IoT) devices. In order to provide this extended flexibility, a middleware was developed to allow for this uniformization.
- Machine Learning Pipeline Integration: This architecture enables us to integrate a standard ML pipeline, namely a training stage based on continuously ingested data and a prediction stage.
- Scalability: The system is container oriented and all data flows go through brokers so that it is horizontally scalable.

Figure 11. Proposed architecture for the general architecture for the graph based framework.



In Figure 11, the framework itself is pictured. Messages from the local MES are injected into a RabbitMQ service and a bridge microservice processes messages and injects them into a Kafka topic. These messages are processed by a middleware which filters out the messages which are not relevant and which translates the relevant ones into the standard data structure for the framework. After this pre-processing, the now standardized message is injected in a distinct Kafka topic where it will be processed by the Graph Metrics module. This module keeps track of all the tokens in the graph, their exact location and the metrics associated with them. In order to store all of this information, the authors rely on an ecosystem of databases for different use cases. For time series, a Prometheus variant was used. As an historical data repository, SQL DB was used but could be replaced by Apache Druid for OLAP. For visualization purposes, the recommended tool is Grafana yet there are more options available, such as Superset or Power BI.

Although this framework does solve the issue of data heterogeneity, it is recommended that organizations follow and implement a concrete roadmap for standardization and modernization of their ISs and manufacturing devices. Even though this is an enormous collective effort, it will certain pay-off in the long term since data heterogeneity involves added risk, higher maintenance costs and added development costs due to all the extra middleware components.

CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

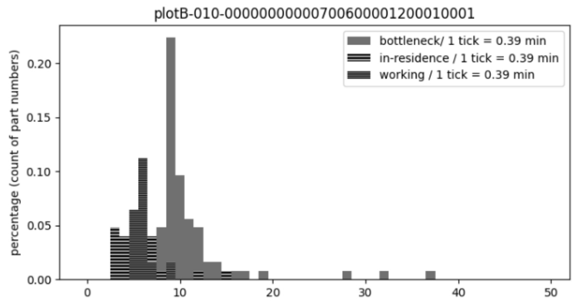
Since the authors recognize the usefulness of the application of the techniques previously mentioned, they intend to make these functionalities readily and easily available to the end user. By disguising all of the inherent complexity, the plan is for the so-called process expert to be able to take the most out of the system in a proactive way. In order to achieve this, there is the need to come up with some creative interface methods such as asking the output results via email using natural language. Some Natural Language Processing methods would then translate the user's inputs into framework action triggers that produce and return reports answering users' questions.

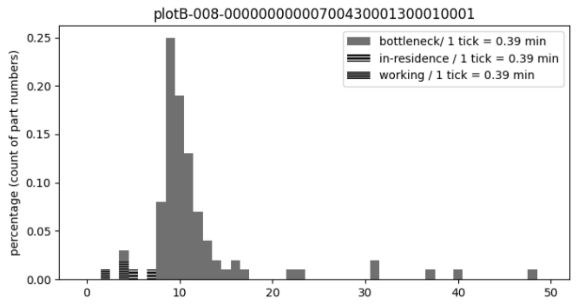
Besides user interaction with the outputs from the system, there are three main other research directions.

One of these would be to use Priority Queues instead of FIFO (First In First Out) as a way of reducing the throughput time in a graph, in this particular case, applied to a PL. In this way, products which are known to have a behavior which is more variable would be processed in a separate way in order to prevent disruption to the graph's overall performance. This method would be first simulated and then applied to a real use case if the results are promising.

To understand the nature of the nodes durations, some additional figures are provided which show this distribution. As one can see from Figure 12 (left), there may be scenarios where the distribution does not really follow a log-normal and has a high number of outliers. However, these stations tend to be the exception rather than the norm. Most stations follow a pattern like the one seen on Figure 12 (right).

Figure 12 (left) and Figure 12 (right). On the left, plot of the duration of processing at Station 101 of the production line. On the right, plot of the duration of processing at Station 43 of the production line.





A second research direction would be to apply this same framework to bottleneck identification and optimization. Bottleneck detection is a methodology which could be easily applied to a framework like this since it takes first order metrics and creates a classification based on these metrics e.g. this station is a bottleneck or not in a given moment in time. The framework already provides the required metrics such as node and transition duration to do this analysis. This is a clear example of the practical usefulness of having a framework which has a stratified structure and where components are built on top of each other.

Finally, Root Cause Analysis for getting to the source of issues in local or overall graph performance can also be achieved through the use of such a framework. One could, for instance, understand why the duration in a certain station has a higher dispersion than all other stations. This could stem from various origins: the intrinsic nature of the process which takes place in a certain node, the interaction of the token with the node or the interaction of a third party with the node, for instance, the operator that may be working on a certain workstation.

In this chapter, the authors have explored a generic methodology to solve a wide range of problems which can be modeled by a graph. The starting hypothesis was whether a framework based on graph systems using AutoML could be efficient in solving problems of the nature already described. Based on the experiments that were run, the authors can conclude that this is a robust and accurate way of introducing ML to solve a panoply of real life problems. Using this methodology means that although there may be a standard toolset, it does not mean, however, that all the problems are solved using the same classifier. The advantage of this methodology is precisely that there is no need to fix the classifier which is being used beforehand. That will depend on the intrinsic nature of the graph and processes which are being modelled. With that said, the pipeline is dynamic and can change over time as conditions also change. With this approach, the best of both worlds is combined to achieve both optimal flexibility and accuracy.

ACKNOWLEDGMENT: The authors acknowledge the support of Bosch Termotecnologia. The present study was also developed in the scope of the Project Augmented Humanity [POCI-01-0247-FEDER-046103], financed by Portugal 2020, under the Competitiveness and Internationalization Operational Program, the Lisbon Regional Operational Program, and by the European Regional Development Fund. The first author was supported by Bosch Termotecnologia. The second author was partially supported by the Center for Research and Development in Mathematics and Applications (CIDMA), through the Portuguese Foundation for Science and Technology, reference UIDB/04106/2020. The third author is funded by National Funds through the FCT - Foundation for Science and Technology, in the context of the project UIDB/00127/2020 (IEETA).

REFERENCES

Backus, P., Janakiram, M., Mowzoon, S., Runger, C. & Bhargava, A. (2006). Factory Cycle-Time Prediction with a Data-Mining Approach. *Semiconductor Manufacturing, IEEE Transactions* on. 19. 252 - 258. 10.1109/TSM.2006.873400.

Buck, J. & Lee, E. Scheduling dynamic dataflow graphs with bounded memory using the token flow model. In *1993 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 1, 1993, 429-432.

Centomo, S., Panado, M. Fummi, F., "Cyber-Physical Systems Integration in a Production Line Simulator," (2018). *2018 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC),* Verona, Italy, 2018, pp. 237-242. 10.1109/VLSI-SoC.2018.8644836.

Chen, H., & Boning, D. (2017). Online and Incremental Machine Learning Approaches for IC Yield Improvement. *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD).*

Columbus, L. (2019). 10 Ways Machine Learning Is Revolutionizing Manufacturing in 2019. https://www.forbes.com/sites/louiscolumbus/2019/08/11/10-ways-machine-learning-is-revolutionizing-manufacturing-in-2019/?sh=7e5167012b40

Domingos, P. (2017). *The Master Algorithm*. Penguin Books.

Eshuis, R., & Wieringa, R. Comparing Petri net and activity diagram variants for workflow modelling – a quest for reactive Petri nets. In H. Ehring, W. Reising, G. Rozenberg, and H. Weber, *Petri Net Technology for Communication-Based Systems*, volume 2472, Springer, Berlin, Heidelberg, 2003.

Haefner, N., Wincent, J. & Gassman, O. (2021). Artificial intelligence and innovation management: A review, framework, and research agenda. *Technological Forecasting and Social Change*, 162.

Hartley, J., & Sawaya, W. (2019). Tortoise, not the hare: Digital transformation of supply chain business processes. *Business Horizons,* 62(6), 707-715.

Lee, I., & Shin, Y. (2020). Machine learning for enterprises: Applications, algorithm selection, and challenges. *Business Horizons*, 63(2), 157-170.

Liu, W., & Chang, C. (2007). Variants of Principal Components Analysis. *2007 IEEE International Geoscience and Remote Sensing Symposium*, 1083-1086, doi: 10.1109/IGARSS.2007.4422989.

Ma, J., Wang, A., Lin, F., Wesarg, S., & Erdt, M. (2019). A novel robust kernel principal component analysis for nonlinear statistical shape modeling from erroneous data. *Computerized Medical Imaging and Graphics*, 77, 101638.

Maaten, L., & Hinton, G. (2008). Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9(86), 2579−2605.

Oosterbaan, R. J. (2019). Software for generalized and composite probability distributions. *International Journal of Mathematical and Computational Methods*, 4, 1-9.

O’Neil, C. (2016). *Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy*. Penguin Books.

Randal S. Olson, Nathan Bartley, Ryan J. Urbanowicz, and Jason H. Moore (2016). Evaluation of a Tree-based Pipeline Optimization Tool for Automating Data Science. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016,* GECCO '16 (pp. 485-492). Association for Computing Machinery, New York, NY, USA.

Russkikh, P. & Kapulin, D. (2020). Simulation modeling for optimal production planning using Tecnomatix software. In *Journal of Physics: Conference Series 1661*, 012188.

Sci-Kit Learn (2020). API Reference. https://scikit-learn.org/stable/modules/classes.html#module-sklearn.preprocessing

Tušar, T., Gantar, K., Koblar, V., Ženko, B. & Filipič, B. (2017). A study of overfitting in optimization of a manufacturing quality control procedure. *Applied Soft Computing*, Volume 59, Pages 77-87.

Weichert, D., Link, P., Stoll, A., Rüping, S., Ihlenfeldt, S., & Wrobel, S. (2019). A review of machine learning for the optimization of production processes. *International Journal of Advanced Manufacturing Technology*, 104(5-8), 1889-1902.

Wu, D., Jennings, C. Terpenny, J., Gao, R. & Kumara, S. (2017). A Comparative Study on Machine Learning Algorithms for Smart Manufacturing: Tool Wear Prediction Using Random Forests. *Journal of Manufacturing Science and Engineering*. 139. 10.1115/1.4036350.

KEY TERMS AND DEFINITIONS

AutoML: The process of automating the process of applying machine learning to real-world problems. AutoML covers the complete pipeline from the raw dataset to the deployable machine learning model.

Bottleneck: A bottleneck is a constraint in a manufacturing line where one single station is setting up the pace of the line. A bottleneck station is generally always working around the clock without any idle time. Bottlenecks are meant to be avoided since they hinder the performance of the overall line.

Graph: An abstract mathematical entity which possesses nodes and transitions. It is used to model systems with interactions between its various nodes. A manufacturing plant or a line are both examples of physical entities that can be described by a graph.

Manufacturing Execution System (MES): An Information System which is meant to control manufacturing operations in real time while also gathering data from the manufacturing process.

Token: An abstract mathematical entity which moves within a graph. It navigates between the graph’s nodes.