2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications

iSTEP, an integrated Self-Tuning Engine for Predictive maintenance in Industry 4.0

Daniele Apiletti†, Claudia Barberis†, Tania Cerquitelli†,
Alberto Macii†, Enrico Macii‡, Massimo Poncino†, and Francesco Ventura†
† *Department of Control and Computer Engineering*
‡ *Interuniversity Department of Regional and Urban Studies and Planning*
Politecnico di Torino, Torino, Italy
{name.surname}@polito.it

***Abstract*—The recent expansion of IoT-enabled (Internet of Things) devices in manufacturing contexts and their subsequent data-driven exploitation paved the way to the advent of the Industry 4.0, promoting a full integration of IT services, smart devices, and control systems with physical objects, their electronics and sensors. The real-time transmission and analysis of collected data from factories has the potential to create manufacturing intelligence, of which predictive maintenance is an expression. Hence the need to design new approaches able to manage not only the data volume, but also the variety and velocity, extracting actual value from the humongous amounts of collected data.**

To this aim, we present iSTEP, an integrated Self-Tuning Engine for Predictive maintenance, based on Big Data technologies and designed for Industry 4.0 applications. The proposed approach targets some of the most common needs of manufacturing enterprises: compatibility with both the on-premises and the in-the-cloud environments, exploitation of reliable and largely supported Big Data platforms, easy deployment through containerized software modules, virtually unlimited horizontal scalability, fault-tolerant self-reconfiguration, flexible yet friendly streaming-KPI computations, and above all, the integrated provisioning of self-tuning machine learning techniques for predictive maintenance.

The current implementation of iSTEP exploits a distributed architecture based on Apache Kafka, Spark Streaming, MLlib, and Cassandra; iSTEP provides (i) a specific feature engineering block aimed at automatically extracting metrics from the production monitoring time series, which improves the predictive performance by 77% on average, and (ii) a self-tuning approach that dynamically selects the best prediction algorithm, which improves the predictive performance up to 60%. The iSTEP engine provides transparent predictive models, able to provide end users with insights into the knowledge learned, and it has been experimentally evaluated on a public unbalanced failure dataset, whose extensive results are discussed in the paper.

***Index Terms*—Predictive maintenance, machine learning.**

I. Introduction

Due to the large amounts of data produced by modern and always connected industries, the necessity of powerful and reliable architectures is becoming prominent. From the smart factory, filled with sensors over the whole production chain, to the more advanced IT industry with huge amounts of log data, it is of paramount importance the ability to collect, manage and elaborate all these data in real time, extracting useful knowledge for production process improvements and competitive business advantages.

The paper presents iSTEP, an integrated Self-Tuning Engine for Predictive maintenance targeting Industry 4.0 contexts. The contributions of the proposed approach consist in the integration of the monitoring and prediction tasks, the introduction of a feature engineering block yielding a large improvement in prediction performance, the addition of a self-tuning approach for the dynamic selection of the best predictive algorithm, and specific attention devoted to the transparency of the predictive models, hence providing interpretable knowledge to end users and competitive business advantage to manufacturing companies.

The paper is organized as follows. Section II presents related work, Section III describes the iSTEP building blocks, Section IV provides the current technological implementation, Section V discusses the experimental results, and Section VI draws conclusions.

II. Related work

A key aspect addressed by the state of the art is the management of business processes and risks. In [1] an approach for a holistic data-driven assessment of risks based on time series is proposed. A case study exploiting Big Data analytics to improve production processes is described in [2]. It is based on the methodology named Cross-Industry Standard Process for Data Mining. It is used to present and organize results for better understanding businesses. Manufacturing computerization is another crucial facet of the Industry 4.0 ecosystem. In [3] a semantic reduction of heterogeneous sources is presented, based on Semantic Web approaches leading to a data integration fostering better analytics implementations. Furthermore, a prediction and detection algorithm based on data analysis for huge amount of low quality data is presented in [4] and applied to traffic congestion prediction and detection in urban area. Additionally, not only the heterogeneity and the quality of data collected in the context of industry 4.0 is a challenge, but also the increasing amount of data to be managed by machine learning techniques; in this context, a comparison between multi-class classifiers and deep learning techniques is presented in [5], and a comparative analysis of exploratory techniques for Big Data is provided in [6]. In [7], authors present a Big-Data scalable predictive approach in the energy

978-1-7281-1141-4/18/$31.00 ©2018 IEEE
DOI 10.1109/BDCloud.2018.00136

domain, a context of crucial importance in the industry 4.0 as well. An energy signature is devised and exploited by means of machine learning techniques to forecast future power consumption. Moreover, in [8] the authors propose a framework for on-demand remote sensing data analysis to accelerate the execution of the models by reducing data transfers through the network allowing classical remote data service systems to evolve into remote sensing data processing infrastructures. Advanced sensors and ICT technologies provide the ability to link physical manufacturing facilities and machines to Internet applications; in [9] a review of virtualized and cloud-based services is provided in the context of manufacturing systems. The paper presents a predictive maintenance approach involving cyber-physical systems with wide Internet of Things capabilities, and complex event processing features.

Among the most widespread policies, Condition-based Maintenance (CBM) is often the most effective. Efficiently determining the health status of a monitored device, in such context, is a crucial goal. Prognostics and diagnostics applied to sensor data aim at determining such system health, by exploiting anomalies in the data. A data-mining-technique overview, among those belonging to the anomaly detection approaches, is provided in [10]. They are devised to exploit artificial neural networks in large systems to effectively predict their health. Prognostics also includes the estimation of the Remaining Useful Life (RUL) indicator. In [11] a deep-belief network ensemble method with multiple objectives is presented for performing RUL estimation. Similarly, in [12] a neural-network prognostics model is proposed to support industrial maintenance scheduling. The failure probabilities are computed from actual equipment measurements by means of a logistic regression approach. Such measurements are then routed to a prognostics model to predict failure conditions and finally to estimate RUL.

A further issue to address is data quality. A method for improving data quality in diagnosing the health of devices and production equipment is proposed in [13]. First, a visualization-based grouping, exploiting the dissimilarity spectrum, is performed on critical measurements, which are then clustered, and finally evaluated in terms of their fitness and separation with each other. An outlier-detection visual assessment is also presented to identify outliers in the data.

The proposed iSTEP engine enhances the state of the art by (i) defining an integrated platform for both the data management and the predictive analytics; (ii) providing a scalable and flexible general-purpose approach, able to fit most Industry 4.0 use cases, as provided by the industrial partners we worked with; (iii) introducing a self-tuning approach for the predictive maintenance block and an automatic feature engineering phase, both centered around human interpretability, thus allowing domain experts to better exploit the machine learning models in their specific context without deep knowledge of the data mining techniques.

III. The iSTEP architecture

Figure 1 depicts the building blocks of the proposed engine and their functional connections. iSTEP (integrated Self-Tuning Engine for Predictive maintenance) exploits data stream processing jointly with machine learning algorithms to perform two main tasks:

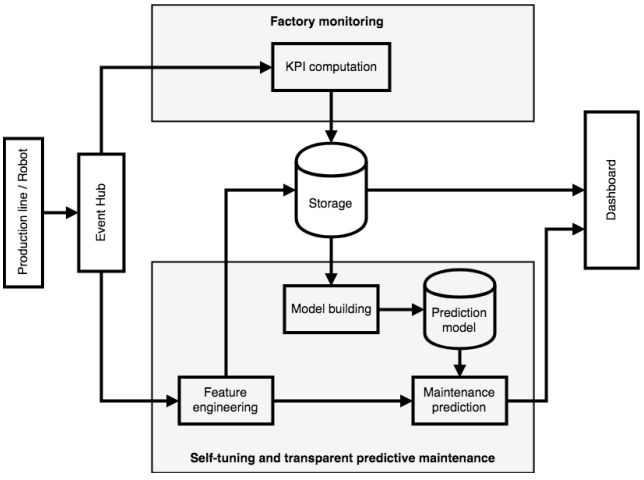
- factory/production-line monitoring, and
- self-tuning predictive maintenance

by collecting, managing, analyzing and visualizing data of interest in an Industry 4.0 manufacturing plant.

The first task aims at generating informative dashboards containing Key Performance Indicators (KPIs) computed in real-time on data collected from production lines. The second task, instead, analyzes manufacturing data streams to predict possible failures, hence preventing production line stops. To this aim, a prediction model is built on historical data by means of machine learning algorithms. iSTEP introduces a self-tuning strategy to automatically configure and select the optimal machine learning algorithm, together with an automatic feature engineering technique.

To actually monitor the whole production process, the data collection block of iSTEP is designed to reliably routing virtually unlimited sensor and log data from heterogeneous sources at different rates. We named such block *event hub*, since, in the predictive maintenance context, the incoming data typically describe events of interest. The event hub collects and routes huge amounts of data in real time avoiding information loss. It is also able to apply basic pre-processing rules, such as filters and test conditions on single events or on small windows of recent buffered data.

Fig. 1: Building blocks of iSTEP



Data from the *event hub* is then routed to the main processing blocks of iSTEP: the factory monitoring and the self-tuning predictive maintenance blocks.

A. Factory monitoring

We identified two operational roles of the iSTEP engine interested in factory monitoring: (i) production managers and

(ii) head of the production shift. Production managers are mainly interested in manufacturing productivity over time with the aim of guaranteeing an optimal level of production. They are also interested in assessing the conceptual and real aspects that may contribute to factory inefficiency as health of devices and production equipment. The heads of the production shift are mainly interested in locally and promptly assessing the overall equipment effectiveness to immediately react to possible sudden issues causing production line inefficiencies.

To this aim, iSTEP provides a real-time monitoring of KPIs such as the OEE (Overall Equipment Effectiveness). OEE is a standard indicator for measuring manufacturing productivity. It identifies the ratio of productive manufacturing time. An OEE score of 100% means that the manufacturing line is producing at maximum speed (100% performance) only good parts (100% quality), with no interruptions (100% availability). OEE allows to uncover insights on how to improve the manufacturing process, hence becoming a best-practice measurement in manufacturing, since it is the single best metric for improving the productivity of manufacturing equipment (i.e., eliminating waste) and identifying losses.

The *KPI computation* block receives data collected by the *event hub* on the overall monitored plant or selected production lines and equipment. Given a buffer with the latest collected data, metrics of interest are derived and computed, updating related dashboards and possibly triggering user-defined alerts.

B. Self-tuning and transparent predictive maintenance

The aim of the predictive maintenance block is two-fold: (i) building a transparent prediction model based on historical data by means of machine learning algorithms, and then (ii) applying such model in real time to new incoming data streams, to identify possible failures. It consists of three parts: (1) feature engineering, (2) model building, and (3) maintenance prediction.

1) feature engineering: The feature engineering block derives relevant static features from the most recent window of the input data time series, supporting the predictive maintenance goal. Among the many choices and challenges we faced in this crucial task, an often under-evaluated requirement is the human-readability of both the extracted features and the predictive model, a requirement also known as transparency. The understanding of the reasons for a possible failure and the link with the originally collected data is of paramount importance for the industries. Apparently in contrast with the academic quest for top prediction accuracy, often industrial partners prefer lower accuracy but more descriptive models, since such property translates into an actionable business advantage with respect to competitors. Hence, the set of static features extracted from a data window (i.e., the time series of the latest N measurements for each variable or a time-based slot of recent data) are the following: noitemsep

- linear regression intercept and slope,
- max value, min value, max absolute value,
- sum of values (the integral, or evidence accumulation),
- sum of values weighted by recency (memory simulation).

Such features are meant to capture the meaning of most signals monitored in Industry 4.0. Since they typically present signs of performance degradation towards the failure event, the feature engineering generates time-series trend indicators such as the slope, and tracks the key values such as min and max. Furthermore, since sequences of events indicating performance degradation tend to be predictors of more probable or imminent failures, we track the sum of the values in the time window to capture such effects.

Results of the feature engineering are saved to the long-term storage database, ready to be used to train new updated models.

2) Model building: On a batch schedule, the *model building* block is executed on historical data. Historical data consist of (i) the original measurements, (ii) the additionally extracted features for each time window, and (iii) the corresponding class labels (failure presence o absence). All data are linked with an object of interest, the device or piece of equipment that can fail and whose predictive maintenance is desired.

The *model building* block executes three preliminary steps, intended to address very common issues in Industry 4.0 contexts.

- Feature filtering, by discarding those having a high number of unacceptable values, e.g., null or missing values, out-of-range measurements.
- Feature ranking, to guide the understanding of feature contributions to the prediction task.
- Down-sampling the non-failure class, since the failure class is almost always under-represented by order of magnitudes.

Feature ranking is approached by computing the correlation matrix of all couples of the original measurement variables, and sorting the features by increasing average correlation score. The hypothesis is that a variable which is loosely correlated with the others, on average, brings new information to the problem, hence it can be useful. On the contrary, many variables are often highly correlated, hence they can be discarded as redundant, simplifying the model training phase.

Down-sampling the non-failure class exploits re-sample without replacement until a selected ratio between the two classes is reached (e.g. 1:10). Finally, only the extracted features and the class labels are used to train the model, discarding the original time-series of measurements, hence enabling the full range of classifiers to be exploited, since most of them do not typically address time series data by design.

The results presented by the system to industrial end-users are two-fold. The self-tuning approach provides an automatic overview of the predictive performance of a wide set of classification algorithms, whose parameters are automatically optimized by a grid-search, hence automatically identifying the best one and its performance gain with respect to the others. Prediction performance is evaluated by exploiting a 5-fold stratified cross-validation, and the training dataset defaults to the full historical data, even if shorter recent periods can be selected to address replacements and upgrades of equipment, devices, or other major production changes. The

trend of the prediction performance (F-measure, precision, and recall of the failure class) is presented for increasing numbers of selected features according to the feature ranking, hence providing insights on the relevance of including such features in the model building phase. All steps are designed to provide an understandable model for humans, also known as model transparency, hence the choice of interpretable machine learning algorithms such as Random Forest (RF) [14], Logistic Regression (LR) [14], Support Vector Machines (SVM) [14], and Gradient-Boosted Tree (GBT) [15]. Furthermore, the performance trend for different number of features is particularly appreciated by industrial partners because drives a better understanding of the phenomena under study.

3) Maintenance prediction: The prediction block applies the prediction model, trained in the model building phase, to new incoming data streams. Incoming data are pre-processed by the feature engineering block. The output is the prediction result, whose aim is to anticipate a failure event in the near future. Results are sent to the dashboard collecting all outputs from both the monitoring and the predictive blocks.

IV. The iSTEP technological solution

To translate the system described in the paper into an actual implementation, we collected requirements during research activities and experiences with real-world industrial contexts, from mechanical to automotive, from automation to food production. In the following, we provide rationale for the design and development choices taken to address the needs of the modern manufacturing industries.

First of all, generally desirable requirements are reliability, availability, scalability, and manageability. Popular technologies exist that reasonably guarantee such properties. The latter requirements are addressed by deploying the whole architecture in a containerized environment, drastically reducing the architectural management complexity, both in development and production phases. Containerized blocks are typically horizontally scalable, as they can be easily instantiated dynamically by design. The possibility of having multiple disposable containers running at the same time also provides availability and reliability at the architectural level, both on-premises and in cloud environments. In our approach, each block is encapsulated into a Docker container [16], hence resulting into a flexible loosely-coupled architecture: the event hub, the factory monitoring block, the predictive maintenance block, the long-term storage, and the dashboard are deployed in separate containers and can be instantiated simultaneously, hence providing horizontal scalability.

The event hub must be able to collect, apply basic preprocessing rules, and route huge amounts of data in real time avoiding information loss. To this aim, we exploit Apache Kafka [17], a horizontally scalable, fault-tolerant, advanced message broker for real-time applications with very high throughput and a long-standing widespread adoption.

Both the factory monitoring and the predictive maintenance blocks are based on Apache Spark [18] and its streaming extensions. Spark Streaming is a state-of-the-art horizontally-scalable high-throughput fault-tolerant framework for soft real-time Big Data analysis. Spark also provides a parallel machine learning library, MLlib [19].

The storage component is built on top of Apache Cassandra, a column oriented NoSQL database. Cassandra is very popular for its throughput scalability [20], especially for write operations, which represent the specific challenge of Industry 4.0 data collection.

V. Experimental results

The experimental section is organized as follows. Section V-A describes the dataset used for the experiments, Section V-B presents the experimental goals and the performance metrics, Section V-C reports the software and hardware setup, Section V-D and Section V-E discuss the prediction performance for different algorithms on the full dataset and on the quarterly datasets respectively, Section V-F evaluates the benefits of the self-tuning block, Section V-G assesses the impact of different number of features, and Section V-H explores the effect of different training sets.

A. Dataset

Experiments have been performed on a public dataset [21] with very large historical data and failure labels. It consists of 90 attributes describing the SMART hard-drive measurements, and corresponding device failures, for 125627 hard disks, over 9 quarters from 2016 to 2018-Q1. The dataset presents 3323 failures among a total number of 63892896 records, hence leading to a very unbalanced failure class representing only 0.005% of the data. Thanks to such extremely unbalanced class representation, the dataset can effectively simulate predictive maintenance scenarios in real-world industry 4.0 contexts. Even if the dataset collects a set of 90 measurements from each device every day, hence the throughput is not properly simulated, such challenge is not addressed by the current experimental session, because the scalability of the Kafka-based event hub and the Apache Streaming framework is widely tested in state-of-the-art publications.

The features selected as non-redundant and less correlated are the 13 attributes reported in Table I. For each feature of those 13, the feature engineering is executed, generating a total of 65 features as input to the model building. Extracted features are computed over windows of 3 months.

B. Goals and performance metrics

Experiments evaluate the capability of the system to support industrial partners in effectively performing the predictive maintenance task, by addressing many of their popular questions:

- Which is the best algorithm to effectively predict failures? How much advantage in performance does the best algorithm provide with respect to the others? Should I change algorithm over time?
- How does the number of features impact on prediction performance? and which are the selected features?

TABLE I: Selected features sorted by correlation score.

feature	correlation score
smart_5_raw	0.002
smart_3_raw	0.053
smart_199_raw	0.079
smart_10_raw	0.082
smart_4_raw	0.082
smart_194_raw	0.095
smart_1_raw	0.101
smart_12_raw	0.106
smart_197_raw	0.136
smart_198_raw	0.136
smart_193_raw	0.145
smart_192_raw	0.163
smart_9_raw	0.184

- How many historical data should be used to train the model? and how does this choice affect the prediction performance?

Prediction performance [14] is defined in terms of precision (i.e., the ratio of true failures with respect to all predicted failures), recall (i.e., the ratio of true predicted failures with respect to all true failures in the test set), and F-measure (also known as F_1 score, i.e., the harmonic mean of precision and recall) of the failure class. High precision indicates that, when a failure is predicted by the model, then it is highly probable to be actually happening. On the contrary, low precision indicates that the model predicts a lot of false failures (type I errors): such events in real-world contexts cause human end users to ignore the model predictions. High recall indicates that the model can correctly predict most failures, or in other words, that few actual failures happen without the model having predicted them. On the contrary, low recall indicates that there are many actual failures happening without the model being able to predict them: such behavior leads to useless predictive maintenance. Depending on the specific application context, precision or recall can assume different importance. However, they are often considered equally important, hence using the F-measure allows to consider both in a single metric.

C. Hardware, software and timing

Experiments have been performed on an Intel Core i7 machine with 32 GB of RAM running Ubuntu 16.04 and a cluster of two nodes configured with Apache Kafka 1.0, Spark (and MLlib) 2.2.0, Docker 17.09, and Cassandra 3.11. Provided such resources, the time required to build the models on the full dataset ranges from the 137 seconds of the LR, to the 307 seconds of the GBT classifier.

D. Algorithm comparison on the full dataset

The algorithms included in the experimental comparison are selected among those available in the Apache Spark MLlib [22] library: Random Forest (RF) [14], Logistic Regression (LR) [14], Support Vector Machines (SVM, specifically the linear SVC variant) [14], and Gradient-Boosted Tree (GBT) [15].

Depending on the specific manufacturing application under monitoring, a prediction performance metric among precision, recall, and F-measure is selected to be optimized by the self-tuning block. Considering a model trained on the overall dataset from 2016 to 2018, evaluating its performance with a 5-fold stratified cross validation, and a down-sampling ratio of 1:10, if the precision metric is selected for the self-tuning block, the RF algorithm is selected by iSTEP to be applied, since it reaches the highest precision value (0.63, see Table II). Instead, GBT would have been selected based on recall (0.82) or F-measure (0.69). A detailed discussion on the performance gain provided by the self-tuning feature of iSTEP is provided in Section V-F.

TABLE II: Prediction performance on the full dataset.

algorithm	precision	recall	F-measure
RF	**0.63**	0.66	0.64
LR	0.55	0.52	0.54
SVM	0.35	0.50	0.41
GBT	0.60	**0.82**	**0.69**

From an operational point of view in the manufacturing context, a recall of 0.82 means that more that 80% of the failures can be predicted. If predicting a failure triggers a maintenance operation preventing a production line stop, then 80% of the costs due to those sudden stops can be saved. Precision is lower than recall, however approximately 60% of the alerts predicting a failure actually corresponds to true failures. This means that if all predicted failures trigger a maintenance intervention, approximately 40% of the costs will be wasted. However, typically the costs for sudden production stops (whose savings are estimated to be 80%) are much higher than maintenance intervention (40% wasted), and the latter are currently scheduled based on time, hence being already performed in excess.

E. Algorithm comparison on quarterly datasets

Fig. 2: F-measure for each algorithm for each quarter.

In manufacturing plants, the production process, line, or equipment often change over time, hence it is important to re-train the model from time to time, so that the model fits the actual context. To better investigate the effect of such re-training, we divided the dataset into quarters and analyzed the performance of each algorithm for each quarter, as if they were separate datasets (5-fold stratified cross-validation for each quarterly dataset). Results are reported in Table III for the F-measure, in Table IV for precision, and in Table V for

TABLE III: Failure class prediction, F-measure.

	Algorithm			
Dataset	GBT	LR	SVM	RF
2016_Q1	**0.74**	0.53	0.48	0.62
2016_Q2	**0.74**	0.56	0.59	0.66
2016_Q3	**0.62**	0.61	0.57	0.59
2016_Q4	**0.68**	0.60	0.49	0.67
2017_Q1	**0.68**	0.50	0.40	0.67
2017_Q2	**0.72**	0.54	0.38	0.71
2017_Q3	0.63	0.53	0.36	**0.67**
2017_Q4	0.55	0.51	0.37	**0.59**
2018_Q1	0.55	0.50	0.36	**0.59**
Average	**0.66**	0.54	0.44	0.64

TABLE IV: Failure class prediction, precision.

	Algorithm			
Dataset	GBT	LR	SVM	RF
2016_Q1	**0.66**	0.54	0.44	0.62
2016_Q2	**0.63**	0.55	0.52	0.58
2016_Q3	0.50	**0.52**	0.45	0.48
2016_Q4	0.57	0.57	0.40	**0.63**
2017_Q1	0.56	0.46	0.31	**0.58**
2017_Q2	0.62	0.64	0.35	**0.70**
2017_Q3	0.53	0.64	0.31	**0.72**
2017_Q4	0.44	0.55	0.30	**0.60**
2018_Q1	0.43	0.54	0.33	**0.68**
Average	0.55	0.56	0.38	**0.62**

TABLE V: Failure class prediction, recall.

	Algorithm			
Dataset	GBT	LR	SVM	RF
2016_Q1	**0.86**	0.53	0.54	0.63
2016_Q2	**0.90**	0.59	0.68	0.78
2016_Q3	**0.82**	0.73	0.80	0.78
2016_Q4	**0.85**	0.65	0.62	0.71
2017_Q1	**0.86**	0.56	0.58	0.80
2017_Q2	**0.87**	0.47	0.44	0.74
2017_Q3	**0.80**	0.45	0.45	0.63
2017_Q4	**0.75**	0.48	0.47	0.60
2018_Q1	**0.76**	0.48	0.43	0.53
Average	**0.83**	0.55	0.56	0.69

TABLE VI: STG comparison for each quarter, F-measure.

dataset	best algorithm	F-measure	STG_max	STG_min
2016_Q1	GBT	0.74	0.26	**0.12**
2016_Q2	GBT	0.74	0.17	0.08
2016_Q3	GBT	0.62	0.04	0.01
2016_Q4	GBT	0.68	0.19	0.01
2017_Q1	GBT	0.68	0.28	0.01
2017_Q2	GBT	0.72	**0.34**	0.01
2017_Q3	RF	0.67	0.31	0.04
2017_Q4	RF	0.59	0.22	0.04
2018_Q1	RF	0.59	0.23	0.04

recall. A better glance of the F-measure results reported in Table III is provided in Figure 2.

We note that different quarters lead to changes in (i) the algorithm performance ranking, with GBT being the best for six quarters, from 2016_Q1 to 2017_Q2, whereas RF is the best for the last 3 quarters, and also in (ii) the variance of the results: the difference between the best and worst result of each algorithm is 0.19 for GBT (mean 0.66), 0.11 for LR (mean 0.54), 0.23 for SVM (mean 0.44), and 0.12 for RF (mean 0.64). RF not only provides generally high performance (ranked second overall), but also the most stable results. SVM, instead, yields to low performance and also highly variable results.

Failures are generally harder to be modeled in recent quarters (2017_Q4 and 2018_Q1), as if the reliability of the hard disks had increased, hence leading to more failures due to random or external causes. In 2017 (Q1-Q2-Q3), with respect to 2016, we notice that the best algorithms (GBT and RF) reach high performance in contrast with a decrease of the worst ones (LR and SVM), whereas in 2016_Q3 almost all algorithms reach very similar performance.

Precision (Table IV) and recall (Table V) results are quite straightforward: GBT is always the best in recall, whereas LR is often the best for precision. The only exceptions are in Q1 and Q2 of 2016, when GBT is the best in both precision and recall, and in Q3, when LR is the best for precision; the latter being the only result with a best algorithm different from GBT and RF.

F. Self-Tuning Gain

Traditional approaches of predictive maintenance tend to perform a thorough study of historical data and then provide a solution consisting of a specific algorithm. However, as we have seen in Section V-E, changes in manufacturing conditions not only require a model re-training over time, but also lead to different best algorithm options. For this reason, we investigate the improvement of dynamically selecting the best algorithm at every model re-training, both on the full dataset and quarterly, thanks to the self-tuning feature of iSTEP. From results reported in Table II on the full dataset, we noticed that, with respect to an a-priori choice of a fixed classification algorithm, the self-tuning approach leads to improvements of up to 0.28 in precision, 0.32 in recall, and 0.28 for the F-measure. We refer to such improvement as STG (Self-Tuning Gain): STG_{min} is the difference between the best performing algorithm and the second best, STG_{max} is the difference between the best performing algorithm and the worst, given the same dataset.

Table VI reports the STG_{min}, STG_{max}, and the best algorithm selected by iSTEP based on the F-measure metric, for each quarterly dataset. Dynamically selecting the best algorithm can provide up to 0.34 improvement in F-Measure performance with respect to the worse algorithm, which is almost 50% of the 0.72 best result (2017_Q2 dataset). However, the worst algorithm used by the STG_{max} is less probably chosen, hence STG_{min} is more meaningful: gains vary from a negligible 0.01 (in 4 quarters) up to 0.12; in 5 quarters STG_{min} is higher than 0.04. Generally, we can conclude that self-tuning the predictive maintenance approach to dynamically select the best algorithm at each re-training is expected to provide tangible savings in costs, as F-measure improvements are in the range 0.04-0.30, which means up to a 60% improvement with respect to the average 0.57 F-measure, across all algorithms.

G. Number of features

The performance trend for different numbers of features is of great interest to industrial partners, since it provides a direct link among the results and the original features contributing to such results, thus disclosing possible interpretations of the failures.

In Figure 3, the F-measure trend is reported considering the best algorithms, GBT and RF, for an increasing number of features[1] as sorted in Table I. Each line corresponds to a different dataset, for which a 5-fold cross validation has been used to compute the resulting F-measure. We selected two full-year datasets, 2016 and 2017, denoted by the non-dashed lines, since they represent the general trend of most quarters. Then we added two specific quarterly datasets, 2017_Q1 and 2018_Q1, denoted by the dashed lines, which showed different behaviors than the average.

Both algorithms present a general behavior (non-dashed lines, yearly datasets) with a two-step improvement: the first noticeable increase in performance is shown at 5 features for GBT and 6 features for RF; then a plateau is present until 8 features for both GBT and RF. Above 8 features GBT slightly increases its performance, whereas RF presents a new plateau until 12 features. Hence, in general, the first 9 features are very relevant for the modeling of the failures in such datasets.

Deviations from the described behavior are reported by the model of the 2017_Q1 failures, reaching its highest results with only 6 features, both for GBT and RF, above which no significant improvement is made. On the contrary, failures in the 2018_Q1 dataset are much harder to model: GBT has an almost linear slope of performance increase, well below the average absolute results, and RF requires at least 9 features, with a reduced 6-feature step.

H. Training dataset comparison

The common academic experimental design to evaluate the prediction model performance consists of a stratified cross-validation of the given dataset into folds of test and training, which is the approach applied in all other experiments of the paper. However, in real-world contexts, the training data set consists of all or part of the historical data, whereas the test dataset is represented by the new incoming data. Hence, we now address the question of which dataset is the best training set for the predictive maintenance goals. To address this issue, we consider two test sets: (i) the worst performing dataset, unanimously for all algorithms, 2018_Q1, and (ii) a relatively easy dataset, 2017_Q1, as discussed in Section V-G. Then we compare the predictive performance by exploiting the two best-performing algorithms (GBT and RF) on two different training sets: (i) the previous quarter and (ii) the previous year, with respect to the selected test set. The previous month is meant to be a model of recent failures, which might be more similar to the new ones. On the contrary, the full past year model collects more data but might include some "old" failures that do not represent the current scenario.

[1]The number of features is considered with respect to the original dataset, before applying the feature engineering.

TABLE VII: Different training sets, GBT algorithm.

Test	Train	Recall	Precision	F-measure
2017_Q1	2016_Q4	**0.85**	0.11	0.19
	2016	0.73	**0.46**	**0.56**
2018_Q1	2017_Q4	0.55	0.20	0.29
	2017	**0.68**	0.20	**0.31**

TABLE VIII: Different training sets, RF algorithm.

Test	Train	Recall	Precision	F-measure
2017_Q1	2016_Q4	**0.59**	0.70	**0.64**
	2016	0.50	**0.86**	0.63
2018_Q1	2017_Q4	0.41	0.70	0.52
	2017	**0.43**	**0.80**	**0.56**

Table VII and Table VIII report results with the GBT and RF classifiers respectively. We notice that there is no clear choice between the year-long dataset or the shorter quarter dataset. GBT yields better results with the larger dataset, whereas RF has a mixed result. In a real-world setting, contextual metadata such as changes in equipment, production line composition, and other structural-changing tasks should be tracked to effectively trigger a model rebuilding. A common trend is the lower precision of smaller datasets., whereas recall and F-measure do not exhibit any specific trend.

VI. Conclusions

The paper presented iSTEP, an integrated Self-Tuning Engine for Predictive maintenance, which is horizontally-scalable, fault-tolerant, and based on containerized environments. Experimental results showed the contributions of specific building blocks to the prediction performance, the trends for different number of features, the performance impact for different datasets and algorithms. The iSTEP engine showed very promising results and stimulated the interest of different manufacturing companies.

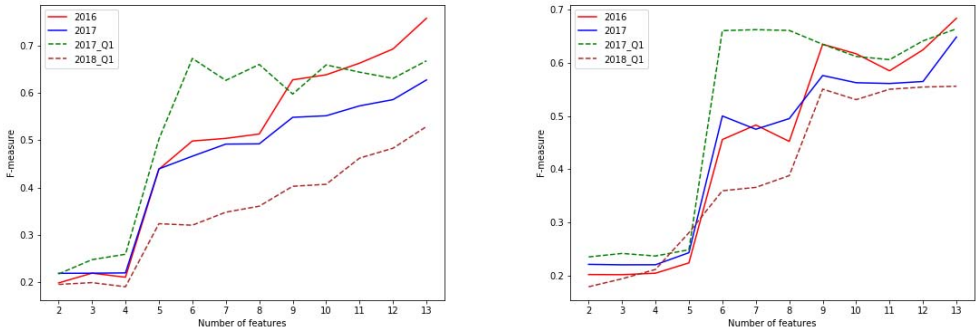
Future works will address (i) the evaluation of different down-sampling ratios for the non-failure class, (ii) improvements to the feature engineering component, e.g., higher-order regressions, and (iii) more complex self-tuning strategies.

Acknowledgment

The research leading to these results has received funding from the following programs.

- POR 2014-2020 of Piedmont Region (Italy), FESR (European Fund of Regional Development), and MIUR (Ministry of Euducation, University and Research, Italy) under the program "Fabbrica Intelligente" (Smart Factory), Action 3, "DISLOMAN" project, Dynamic Integrated Shop fLoor Operation MANagement for industry 4.0.
- European Commission under the H2020-IND-CE-2016-17 program, FOF-09-2017, Grant agreement no. 767561 "SERENA" project, VerSatilE plug-and-play platform enabling REmote predictive mainteNAnce.

Fig. 3: F-measure for different numbers of features, GBT (left) and RF (right) algorithms.



REFERENCES

[1] T. Niesen, C. Houy, P. Fettke, and P. Loos, "Towards an integrative big data analysis framework for data-driven risk management in industry 4.0," in *2016 49th Hawaii International Conference on System Sciences (HICSS)*, Jan 2016, pp. 5065–5074.

[2] M. Nio, J. M. Blanco, and A. Illarramendi, "Business understanding, challenges and issues of big data analytics for the servitization of a capital equipment manufacturer," in *2015 IEEE International Conference on Big Data (Big Data)*, Oct 2015, pp. 1368–1377.

[3] V. Jirkovsky, M. Obitko, and V. Marik, "Understanding data heterogeneity in the context of cyber-physical systems integration," *IEEE Transactions on Industrial Informatics*, vol. PP, no. 99, pp. 1–1, 2016.

[4] P. Jiang, L. Liu, L. Cui, H. Li, and Y. Shi, "Congestion prediction of urban traffic employing srbdp," in *2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC)*, Dec 2017, pp. 1099–1106.

[5] M. Mikuf and I. Zolotov, "Comparison between multi-class classifiers and deep learning with focus on industry 4.0," in *2016 Cybernetics Informatics (K I)*, Feb 2016, pp. 1–5.

[6] D. Apiletti, E. Baralis, T. Cerquitelli, P. Garza, F. Pulvirenti, and L. Venturini, "Frequent itemsets mining for big data: a comparative analysis," *Big Data Research*, vol. 9, pp. 67–83, 2017.

[7] A. Acquaviva, D. Apiletti, A. Attanasio, E. Baralis, L. Bottaccioli, F. B. Castagnetti, T. Cerquitelli, S. Chiusano, E. Macii, D. Martellacci, and E. Patti, "Energy signature analysis: Knowledge at your fingertips," in *2015 IEEE International Congress on Big Data*, June 2015, pp. 543–550.

[8] Z. Huang, A. Zhong, and G. Li, "On-demand processing for remote sensing big data analysis," in *2017 IEEE International Symposium on Parallel and Distributed Processing with Applications and 2017 IEEE International Conference on Ubiquitous Computing and Communications (ISPA/IUCC)*, Dec 2017, pp. 1241–1245.

[9] R. F. Babiceanu and R. Seker, "Big data and virtualization for manufacturing cyber-physical systems: A survey of the current status and future outlook," *Computers in Industry*, vol. 81, pp. 128 – 137, 2016, emerging {ICT} concepts for smart, safe and sustainable industrial systems. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0166361516300471

[10] J. Murphree, "Machine learning anomaly detection in large systems," in *2016 IEEE AUTOTESTCON*, Sept 2016, pp. 1–9.

[11] C. Zhang, P. Lim, A. K. Qin, and K. C. Tan, "Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics," *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, no. 99, pp. 1–13, 2016.

[12] S. A. Asmai, A. S. H. Basari, A. S. Shibghatullah, N. K. Ibrahim, and B. Hussin, "Neural network prognostics model for industrial equipment maintenance," in *2011 11th International Conference on Hybrid Intelligent Systems (HIS)*, Dec 2011, pp. 635–640.

[13] Y. Chen, F. Zhu, and J. Lee, "Data quality evaluation and improvement for prognostic modeling using visual assessment based data partitioning method," *Computers in Industry*, vol. 64, no. 3, pp. 214 – 225, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0166361512001753

[14] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*. Springer, 2013, vol. 112.

[15] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: data mining, inference and prediction*, 2nd ed. Springer.

[16] D. Merkel, "Docker: Lightweight linux containers for consistent development and deployment," *Linux J.*, vol. 2014, no. 239, Mar. 2014. [Online]. Available: http://dl.acm.org/citation.cfm?id=2600239.2600241

[17] J. Kreps, N. Narkhede, and J. Rao, "Kafka: a distributed messaging system for log processing," 2011.

[18] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *NSDI'12*, 2012.

[19] A. Spark, "The Apache Spark scalable machine learning library. Available: https://spark.apache.org/mllib/. Last access on May 2018," 2018.

[20] "Benchmarking top nosql databases: Apache cassandra, couchbase, hbase, and mongodb," End Point Corporation, Park Avenue South, New York, Tech. Rep.

[21] "backblaze hard-drive-test-data," https://www.backblaze.com/b2/hard-drive-test-data.html, accessed: 2018-05-23.

[22] X. Meng, J. Bradley, B. Yavuz, E. Sparks, S. Venkataraman, D. Liu, J. Freeman, D. Tsai, M. Amde, S. Owen, D. Xin, R. Xin, M. J. Franklin, R. Zadeh, M. Zaharia, and A. Talwalkar, "Mllib: Machine learning in apache spark," *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 1235–1241, Jan. 2016. [Online]. Available: http://dl.acm.org/citation.cfm?id=2946645.2946679