Journal of Big Data

RESEARCH

Open Access

Using meta-learning for automated algorithms selection and configuration: an experimental framework for industrial big data

Moncef Garouani[1,2,3*], Adeel Ahmad[1], Mourad Bouneffa[1], Mohamed Hamlich[2], Gregory Bourguin[1] and Arnaud Lewandowski[1]

*Correspondence:
moncef.garouani@etu.univ-littoral.fr
[1] UR 4491, LISIC, Laboratoire d'Informatique Signal et Image de la Cote d'Opale, Univ. Littoral Côte d'Opale, 62100 Calais, France
Full list of author information is available at the end of the article

Abstract

Advanced analytics are fundamental to transform large manufacturing data into resourceful knowledge for various purposes. In its very nature, such "industrial big data" can relay its usefulness to reach further utilitarian applications. In this context, Machine Learning (ML) is among the major predictive modeling approaches that can enable manufacturing researchers and practitioners to improve the product quality and achieve resource efficiency by exploiting large amounts of data (which is collected during manufacturing process). However, disposing ML algorithms is a challenging task for manufacturing industrial actors due to the prior specification of one or more algorithms hyperparameters (HPs) and their values. Moreover, manufacturing industrial actors often lack the technical expertise to apply advanced analytics. Consequently, it necessitates frequent consultations with data scientists; but such collaborations tends to cost the delays, which can generate the risks such as human-resource bottlenecks. As the complexity of these tasks increases, so does the demand for support solutions. In response, the field of automated ML (AutoML) is a data mining-based formalism that aims to reduce human effort and speedup the development cycle through automation. In this regard, existing approaches include evolutionary algorithms, Bayesian optimization, and reinforcement learning. These approaches mainly focus on providing the user assistance by automating the partial or entire data analysis process, but they provide very limited details concerning their impact on the analysis. The major goal of these conventional approaches has been generally focused on the performance factors, while the other important and even crucial aspects such as computational complexity are rather omitted. Therefore, in this paper, we present a novel meta-learning based approach to automate ML predictive models built over the industrial big data. The approach is leveraged with development of, AMLBID, an Automated ML tool for Big Industrial Data analyses. It attempts to support the manufacturing engineers and researchers who presumably have meager skills to carry out the advanced analytics. The empirical results show that AMLBID surpasses the state-of-the-art approaches and could retrieve the usefulness of large manufacturing data to prosper the research in manufacturing domain and improve the use of predictive models instead of precluding their outcomes.

Keywords: Algorithms selection, Machine learning, AutoML, Meta-learning, Industry 4.0, Big industrial data, Decision support systems

Introduction

The fourth industrial revolution or Industry 4.0 field is increasingly relying on machine learning algorithms. These are generally used to conduct predictive analytics and gain greater insights into the complex manufacturing processes [1]. For example, ML algorithms have been applied with great success at the process, machine, shop floor and supply chain levels, and have proven effectiveness in predictive maintenance field by predicting the occurrence and severity of machinery failures [2, 3]. Recently, a predictive model [4] based on machine learning has been used to estimate and predict the gradual degradation of such machinery, allowing the operators to make informed decisions regarding maintenance operations. These results, among others, indicate heavy interest in ML development and analysis for manufacturing applications.

As machine learning has proven its benefits and efficiency in many fields, its successful implementation in the context of manufacturing industry requires a large effort from human experts and practitioners since there is no *one size fits all* algorithm that can perform well on all possible problems [5]. Even though being familiar with manufacturing data, industrial researchers are still lacking the machine learning expertise required to handle these industrial big data sources [6].

To overcome such a lack, they often cooperate with data science experts. Nevertheless, for this interactive process to converge, a lot of effort and time is required from both sides. This is due to the fact that devising and deploying ML solutions needs to be started from scratch. This long journey has to start from a lengthy data provisioning process. It continues with finding the right collaborators which requires a continuous back-and-forth exchange between ML experts and industrial actors. Hence, automating activities often require human expertise that would allow smart factories actors and researchers to rapidly build, validate, and deploy machine learning solutions.

Motivated by this goal, Automated Machine Learning (AutoML) [7] has emerged as a new research field aiming to automatically select, compose, and parameterize machine learning models, which are able to achieve an optimal performance on a given task. Owing to the immense potential of AutoML, various learning paradigms have been applied to this task and tools are available to the research community such as TPOT [8], AutoWEKA [9], Auto-sklearn [10], as well as commercially ones such as RapidMiner [11], Big ML [12], and Data Robot [13]. However, the computational complexity, the huge time required to get the results, and the technical expertise required to deploy them, have prevented many applications of these techniques in the manufacturing field [14, 15]. Hence, the industrial needs are yet to be fulfilled [6, 15].

In this paper, we present AMLBID, a novel meta-learning based tool with the advantage of a computational complexity near O(1), intended for automating the selection and parametrization of ML models that deal with industrial big data. Therefore, the proposed solution may improve their quality of service, productivity, and more importantly, reduce the need for ML human experts.

Given a dataset, and an evaluation metric (e.g., predictive accuracy, precision, recall), AMLBID produces a ranked list of all candidate pipelines based on their expected performance with respect to the desired metric. This list is produced based on a meta-knowledge base gained from previously analyzed manufacturing datasets and combinations of pipelines, without executing individual candidate pipelines. We compare the performance of AMLBID to those of current state-of-the-art pipeline generation approaches [8, 10]. The results of the evaluation, conducted on real-world manufacturing classification problems, show that our system achieves comparatively better results to the baselines in terms of predictive performance and run-time.

The remainder of this paper is structured as follows : in "Related work" section, we discuss the closely related works with respect to ML-based data analytics in industry 4.0 and discuss most relevant works on the Automated Machine Learning. "AutoML in manufacturing industry" section briefly review the use of AutoML technology in the manufacturing industry. In "Framework and methodology" section, we provide an overview of our Meta-Learning (MtL) based framework and methodology for supporting the model selection and parametrization in the AutoML process and present its core features. In addition, we discuss in detail the materialization of our proposed tool in terms of a prototype validation solution. In "Evaluation results and discussion" section, we show the results of the empirical evaluations. Finally, "Conclusion and future work" section summarizes the paper and outlines the future perspectives.

Related work

Industrial data analytics

This section presents an overlapped overview of advanced data analytics applications in the manufacturing industry. Various manufacturing levels including process, machine, shop floor and supply chain levels are examined to assess the internal mechanisms of advanced analytics techniques which are applied across different contexts to understand the nature of machine states and anomalies.

Advanced analytics practices at the process level

The process level has a wide-range of advanced analytics applications. Researchers have applied machine learning techniques to improve the products quality control and increase the efficiency of manufacturing processes [14, 16].

Predicting the product quality helps manufacturing engineers to anticipate faulty products and improve the products design. Asif et al. [17] present a machine learning based approach to predict different weld qualities in real-time using weld input parameters and *Acoustic Emission* to overcome the post production evaluation stage that often results in disposal of expensive material or lengthy repair processes. The *Logistic Regression (LR)* and *Sequence Tagging* algorithms are used to predict the presence of five weld states *good, excessive penetration, burn-through, porosity and porosity-excessive* as penetration for decision-making. Their prediction accuracy of *LR* and *Sequence Tagging* algorithms has been of 82.35% and 91.18%, respectively.

Similarly, aiming to classify castings of steel wires for tire reinforcement depending on the number and properties of non-metallic inclusions, Cuartas et al. [18] conducted a study based on 855 observations obtained from the quality control of the steel. Their comparative study on Logistic Regression, K-Nearest Neighbors, Support Vector Classifier, Random Forests, AdaBoost, Gradient Boosting and Artificial Neural Networks algorithms forecasts whether the casting will be rejected or not. It qualifies Random Forest as the most successful algorithm providing an Area Under the Curve (AUC) in the test set of 85%.

Advanced analytics practices at the machine level

In some research works, the machine learning applications are deployed to monitor and understand machine behaviors. Tools conditions, for example, have been monitored using machine learning techniques. Monitoring tools conditions involves the tracking of the evolution of the tools states and detect a fault or breakage [19–21]. Existing literature features the applications of Support Vector Machines (SVM) for the condition monitoring of tools. Medina et al. [22] proposed a machine learning based approach for fault classification in two mechanical equipment (i.e. gearbox and roller bearings). The faults classification is obtained by using a multi-class SVM. The proposed approach has been tested using a 10-fold cross-validation strategy on the vibration signals of these equipment. Their final results show that the proposed approach could achieve a classification accuracy of 99.3% for the gearbox dataset and 100% for the roller bearings. Similarly, an adapted deep neural network strategy [19] has been proposed for condition monitoring of an in-wheel motor (classification of the wear of inner race and outer race). The classification results of the approach achieved a classification accuracy of 99.8%.

Advanced analytics practices at the shop floor and supply chain levels

The advanced analytics are also applied in the manufacturing industry at a higher level. It is aimed at the correct production planning and control which can lead towards the global improvement in manufacturing production systems [23].

The utility of applied analytics has also proven to resolve the supply chain problems which are often complex np-hard combinatorial optimisation problems. Carbonneau et al. [24] used Neural Networks (NNs) to forecast demands in a supply chain to optimize the polynomial time costs and resource usage to fulfill the orders. They compared this technique with regular regression and SVM, concluding that SVMs and NNs are faster, but not more accurate than regular regression models. Likewise, Wu [25] proposed a hybrid intelligent system combining the wavelet support vector machine and particle swarm optimization for forecasting car sales. The obtained simulation results demonstrate the proposed approach as an effective solution in dealing with uncertain data and finite samples.

Common practices to apply advanced analytics for manufacturing-related problems

Predictive analytics have gained significant interest among the industry 4.0 community. Machine learning based data analytics techniques are widely applied across different levels of the manufacturing industry. Wuest et al. [26] summarized the ability of machine learning techniques to meet manufacturing requirements. According to their work, expressively not every machine learning technique is applicable to every manufacturing problem. As manufacturing stakeholders do not possess the necessary expertise to achieve these tasks, they often collaborate with data scientists who may provide guidelines for applying machine learning techniques. In most cases, these collaborations are complex causing excessive consumption of time and effort.

Capabilities to perform advanced analytics without strong data science knowledge are highly desired to facilitate the application of advanced analytics in manufacturing. In our previous work [6], we presented an overview of a manufacturing-specific framework to equip the manufacturing researchers and actors with these capabilities. We discussed different requirements and challenges to build such a system while in this paper, we present a prototype validation tool in order to automate the selection and parametrization of machine learning models and the implementation of the framework to resolve the manufacturing-related problems.

Automated machine learning

The Automated Machine Learning is the process of applying the machine learning techniques, without special assistance or intervention, on the real-world problems [27]. The goal of this research field is usually to enable non-ML experts to effectively utilize "off-the-shelf" solutions and thus save the time and effort for knowledgeable practitioners. AutoML systems follow the no free lunch theorem as "*no single solution fits best for every domain and use case*" [5]. At its core, the AutoML attempts to solve the problem as follows : given a dataset, a machine learning task (e.g. classification, regression, clustering) and a performance criterion (e.g. accuracy, recall, F1 score), perform the task based on the dataset while optimizing the performance criterion [28].

A number of approaches have been proposed to support the machine learning automation. These approaches range from automatic data pre-processing [29] to automatic model selection [30, 31]. Some approaches [8, 9] attempt to automatically and simultaneously select the right learning algorithm and find the optimal configuration of its hyperparameters. These approaches are also referred as Combined Algorithm Selection and Hyperparameters optimization problem (CASH).

Owing to the immense potential of AutoML, a number of different learning paradigms have been applied to this task. Similarly, several tools are available to the research community such as Auto-sklearn [10], AutoWEKA [9], TPOT [8] as well as commercially ones such as RapidMiner [11], and Google AutoML [32]. An ongoing competition [33] around this goal has been running since 2015 focusing on various budget-limited tasks for supervised learning. A brief comparison among the most popular AutoML systems and platforms, in terms of cost, coding requirements, processing location, input data requirements, and supported Operating Systems is given in Table 1.

Auto-WEKA [9] is an AutoML framework with ongoing improvements [36] for building the machine learning pipelines based on the Weka [37] ML library. Auto-Weka addresses the CASH problem using the Bayesian optimization.

Auto-Sklearn [10] is an AutoML toolkit implemented on top of the Scikit-Learn[1] data-mining library. It uses the meta-learning, together with the ensemble construction and Bayesian optimization search procedures to address the CASH problem.

[1] https://scikit-learn.org.

Table 1 Summary of related AutoML systems

System	Cost	Coding need	Data type	Operating system		
				Linux	Mac	Windows
Google AutoML	Billable	No	Tabular images, text	Cloud computing		
Amazon AutoML	Billable	No	Tabular images, text	Cloud computing		
Microsoft AutoML	Billable	No	Tabular images, text	Cloud computing		
Auto-Sklearn [10]	Free	Yes	Tabular	Yes	No	No
ATM [34]	Free	Yes	Tabular	Yes	–	–
TPOT [35]	Free	Yes	Tabular	Yes	Yes	Yes
Auto-WEKA [36]	Free	Yes	Tabular	Yes	Yes	Yes

The Tree-based Pipeline Optimization Tool (TPOT) [8] employs the genetic programming algorithms to optimize ML pipelines by exploring many different possible pipelines. Each pipeline consists of a machine learning model and their hyperparameters configuration. While their evolutionary strategy can cope with this irregular search space, many of the randomly assembled candidate pipelines evaluated by TPOT end up as invalid. It thus result in wasting the valuable time that could have been spent for training the valid models.

Among the big market actors, Google Cloud Platform recently released the AutoML Tables[2], a supervised learning service that handles end-to-end AutoML, but it is only available on Cloud as a managed service of the commercial framework.

While all of these tools provide partial or complete ML process automation, each one works differently, and targets different dataset structures, platforms, algorithms, or end-users, thus posing unique advantages and disadvantages at the same time. For instance, Auto-Sklearn is embedded in Python, however, it only operates on structured data using *Linux Operating System*. Auto-WEKA supports Weka ML algorithms with the advantage of a graphical user interface, but it is limited to statistical algorithms. RapidMiner is among the tools that provide features engineering capability but it requires expert guidance. Whereas, Google AutoML supports most datasets and algorithms but the service is only cloud-based, and it is mostly commercial for dedicated data processing.

AutoML in manufacturing industry

Although AutoML has been applied to a range of purposes and applications [6], few attempts have been made to apply these techniques in the manufacturing field. It is largely observed that the industrial needs are yet to be satisfied as the industrial actors mostly use traditional ML processes rather than AutoML [38].

As mentioned in [6], there are several challenges that tackle the application of machine learning in the manufacturing space. One of the main challenges is concerned with the construction of a high-quality and representative data set. This is a difficult task, as in a manufacturing unit, data are oftenly in heterogeneous formats and each process generates different amounts of data in various formats (CPS data, images, text etc.) along with different quality levels (data variety/skew) [39–41]. Another reason for

[2] https://cloud.google.com/automl-tables.

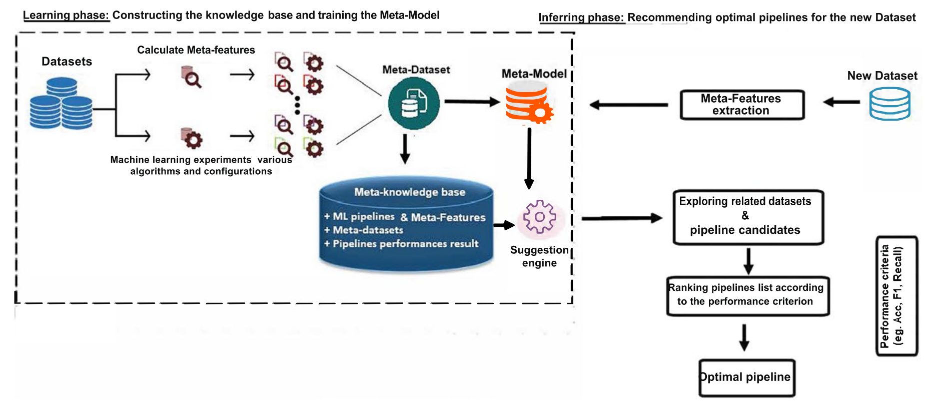
Learning phase: Constructing the knowledge base and training the Meta-Model
Inferring phase: Recommending optimal pipelines for the new Dataset
Datasets
Calculate Meta-features
Machine learning experiments various algorithms and configurations
Meta-Dataset
Meta-Model
Meta-knowledge base
+ ML pipelines & Meta-Features
+ Meta-datasets
+ Pipelines performances result
Suggestion engine
Meta-Features extraction
New Dataset
Exploring related datasets & pipeline candidates
Ranking pipelines list according to the performance criterion
Optimal pipeline
Performance criteria (eg. Acc, F1, Recall)

Fig. 1 The functional architecture and process flowchart of AMLBID

the low adoption rate of AutoML solutions in the industrial space is that current methods for the ML pipeline optimization are inefficient on the large datasets derived from the manufacturing environment.

In order to overcome some of these issues, Lechevalier et al. [14] proposed a framework for semi-automatic generation of analytical models in manufacturing. They also proposed a proof-of-concept prototype that allows practitioners to generate artificial neural networks for prediction tasks through a user interface. Similarly, Villanueva Zacarias et al. [15] present a framework that automatically recommends suitable analytics techniques with respect to a domain-specific problem at hand. Both frameworks have represented promising approaches to tackle the problem of automated-analytics technique configuration in the manufacturing domain. However, these frameworks do not achieve the required goal of identifying the promising combinations of analytics and the application areas in the first place. Therefore, they cannot be used as decision-making tools at the managerial level [42].

Framework and methodology

Meta-learning or learn to learn is a general approach used for predicting how an algorithm will perform on a given task. It is a method that aims at finding the correlation between datasets meta-features (characteristics) and learning algorithms. Given the characteristics of a dataset, a predictive meta-model can be used to forecast the performance of particular ML pipelines. As discussed earlier, the ability to implement knowledge extraction using ML techniques tend to be increasingly complex and time consuming for non-expert users. AMLBID, in this respect, is a meta-learning based AutoML system in the form of a Python package. It attempts to automate the process of the algorithms selection and the tuning of hyperparameters in supervised ML.

Conceptual description

The global architecture of the proposed framework is depicted in Fig. 1. The algorithms recommendation and parametrization is provided by the *Suggestion Engine* through the use of a collaborative knowledge base (KB) which is an inherent part of the system. The KB is simply a collection of inductive meta-features that describe the datasets, the pipelines and their inter-dependencies. Whenever a new dataset is presented to the system, the suggestion engine provide a suggestion of the most appropriate pipelines (classifiers with their hyperparameters configuration). AMLBID has been developed on the meta-learning concept, consequently, it consists of two main phases which are the *learning* phase and the *recommendation* one.

In the **learning phase**, a meta-learning space is established using meta-data about prior learning tasks and previously learned models. It consists of datasets characteristics (*meta-features*) and the performance measures for 08 machine learning algorithms on those particular datasets. The first step is to create a meta-dataset for the selection of a set of datasets to perform machine learning experiments. In the second step, two actions have to be performed on each dataset. On the one hand, the meta-features (characteristics) of the dataset must be calculated. Meta-features describe the dataset in various ways. For instance, it includes the number of attributes, instances and classes in the dataset, the skewness of numerical features. More information about the various meta-features is described in the next section. On the other hand, machine learning experiments have to be run on the datasets. Following these requirements, the results of this step are combined into a single meta-dataset, and as final result of the MtL process, a meta-model is induced. This meta-model represents the mapping between meta-features describing the dataset, and the predictive performance obtained by the group of learning algorithms when applied to these datasets.

The **recommendation phase** is initiated when a new dataset to be analyzed arrives. At this point, a set of meta-features describing the dataset are extracted and fed to the meta-model for the suggestion of the top modeling pipeline(s). Existing automated machine learning tools are not tailored to industrial practitioners and researchers' needs [6, 42]. This work addresses the gaps in existing tools and it can be considered innovative for the following reasons:

1. We present the pioneer framework system that is capable to automate ML predictive models building and configuration over industrial big data. The software may enable manufacturing actors to rapidly build well performing predictive models to improve outcomes and reduce costs for various tasks such as machine's failures prediction as well as the need of classical collaborations.
2. We present a tool to automatically select ML algorithms and hyperparameters configuration for a given machine learning problem which is comparatively more quick than the currently available methods with a computational complexity near *O(1).*
3. We develop an open source python package that can be used by engineers, researchers and data analysts to automatically generate well performing ML pipelines.

Prototypical implementation

We attempt a novel tool to assist the industrial actors during the selection and configuration process of empirically most suitable ML model. For this purpose we adapted the meta-learning paradigm along with the meta-data which are stored in form of a

Table 2 Statistics about the used datasets according to the number of classes, predictive attributes and instances

	Classes	Attributes	Instances
Min	2	5	1800
Max	18	1000	105908

Table 3 Statistics about the used datasets according to related tasks

Task	# Dataset	μ Attributes	μ Instances	μ Classes
Process level	78	29	30529	3
Machine level	248	53	13942	2
supply chain level	74	17	21726	2

The "#" symbol denotes "number of", and "μ" denotes "average of"

meta-knowledge base. It principally maintains the results of different classification algorithms on datasets and meta-features of datasets. In this context, an adequate ML pipeline for a fresh dataset can be selected with the help of the past training experiences. For selection, different distance functions are available to match meta-features of the freshly introduced problem with ones that are available in the meta-knowledge base. In the following section, we discuss in detail the used datasets and their meta-features along with the performance evaluation and the meta-knowledge base construction.

Datasets

In this study, we used 400 real-world manufacturing classification datasets that have been collected from the popular Kaggle,[3] KEEL,[4] UCI,[5] and OpenML[6] platforms. These datasets represent a mix of binary (76%) and multiclass (24%) classification tasks, which are highly diverse in terms of dimensionality, and class imbalance. The meta-features of these datasets are summarized in Table 2.

It is worth noting that the used datasets cover a broad range of application areas, including process level studies, machine related problems and supply chain level, among others (see Table 3). In order to ensure the fairness in our performance comparison, we have not performed any preprocessing operation on the datasets to avoid any potential bias or impact on the classifiers performances.

Meta-features

A meta-feature, also considered as a characterization measure, is a function that extracts relevant characteristics from a dataset to characterize its complexity. The description of a dataset by a set of meta-features produces a numerical values vector. During the

[3] https://www.kaggle.com/.
[4] https://sci2s.ugr.es/keel/datasets.php.
[5] https://archive.ics.uci.edu/.
[6] https://www.openml.org/.

Table 4 A sample list of meta-features used in current study

Meta-feature	Description
Nbr_Instances	Number of dataset instances
Nbr_Classes	Number of classes
Class_Entropy	Class entropy
Nbr_Features	Number of features
Skew_min/max/mean	Min, max and mean skewness of numerical features
Kurtosis_min/max/mean	Min, max and mean kurtosis of numerical features
Class/Attr kurtosis	The distribution probability of classes/attributes

current study, we considered 42 meta-features from each dataset (Table 4). These meta-features can be organized into three main classes :

General measures that include general information related to the problem (dataset) at hand. To an extent, their purpose is to measure the complexity of the underlying dataset (e.g. number of instances, dataset dimensionality, ratio of missing values, attributes and classes, etc.).

Statistical and information-theoretic measures that describe the numerical properties of data distribution in a dataset sample. They include different summary statistics per attribute like mean, standard deviation, etc.

Landmarking that characterize the predictive problems (datasets) when basic machine learning algorithms (with default hyperparameters configuration) are performed on them. In our system, the K-Nearest Neighbor (KNN), Gaussian Naive Bayes (GNB), Decision Trees (DT), and the Linear Discriminant Analysis (LDA) were used as landmarkers.

Pipelines generation

We used 08 classifiers from the popular Python-based machine learning library, Scikit-learn in order to build the meta-knowledge base. These classifiers are *Support Vector Machines*, *Logistic Regression*, *Decision Tree*, *Random Forest*, *Extra Trees*, *Gradient Boosting*, *AdaBoost*, and *Stochastic Gradient Descent* classifier.

To better understand the execution scenario of the proposed framework, the algorithm 1 shows the construction process of the knowledge base. We actually generate at least 1000 different combinations of the hyperparameters configurations for every single execution of a classifier over each dataset. This execution process results in an average of 8000 pipelines for each dataset. It might be useful to note that during the training of this pipeline, we use a 5-fold stratified cross-validation strategy to construct the meta-datasets. As a result, the knowledge base consist of more than 4 millions evaluated classification pipelines. It can be observed that the number of configurations/evaluations of any considered algorithm is not the same due to the different variations of algorithms hyperparameters.

Algorithm 1 Establish the meta-knowledge base.

```
1: Input: $ClassificationAlgs[..]$,                          ▷ available classification algorithms
   $HpSpace[..]$,                                            ▷ set of HPs configurations to be applied
   $PerfMeasures[..]$                                        ▷ set of performance measures to acquire
2: Output: meta_KB[#measure][#metadata]                      ▷ meta-knowledge base
3: function CREATEMETAKB(datasets[ ])
4:     $metadata[\,] = \varnothing$
5:     for each measure $in\ PerfMeasures$ do
6:         for each dataset DS $in\ datasets$ do
7:             $ds\_mf = ComputeMetaFeatures(\mathtt{DS})$;
8:             for each algorithm Alg $in\ ClassAlgs$ do
9:                 for each hyperparameters_configuration Hp $in\ HpSpace$ do
10:                    $ds\_pm = GetPerformanceWith5FoldCV(\mathtt{Alg}, \mathtt{Hp}, \mathtt{DS})$;
11:                    $metadata[\,] \leftarrow ds\_mf \cup ds\_pm$;
12:        $meta\_ds[\mathtt{measure}] \leftarrow metadata[\,]$;
13:    return $meta\_KB$
```

Meta-model

In general, during an AutoML optimization process, the main goal is to efficiently reduce the search space in order to make an effective use for allocated time and resource budget. In particular, it is significant for the optimization process to select, or at least start with, only few classifiers or pipelines, which have the highest potential to provide the optimal performance on the input dataset.

In order to tackle this challenge, we develop a Meta-Model for the prediction of the most appropriate classifier(s) for a given dataset as well as combination of the related hyperparameters configuration. It is based on meta-features, given the fact that we already have the previous knowledge regarding the performance of the classifiers on datasets with similar characteristics.

Let us consider an execution flow in order to better understand the in-depth detail of the functional construction of meta-model. The training process of the meta-model is initiated, for each considered performance measure (i.e., accuracy, precision, recall, and F1-score). We retrieve the set of all the best predicted results of all considered evaluation metrics in compliance to the rule given in algorithm 2.

Algorithm 2 The functional construction of meta-model.

```
1: D: set of training datasets
2: P: set of evaluated pipelines
3: for each $d \in \mathcal{D}$ do
4:     for each $p \in \mathcal{P}$ do
5:         if $performance\_result(p) = max(performances)$ then
6:             $p \leftarrow class1$
7:         else
8:             $p \leftarrow class0$
```

It can be observed that in the case of the best prediction result for a given dataset if the result is above the maximum accuracy achieved by all other classifiers for each dataset then we denote the identified classifier as the top performer classifier (*Class 1*), otherwise we denote the classifier as the low performer classifier (*Class 0*) for the concerned dataset. In other words, the *Class 1* label reflects that the classifier has the potential to be among the best performing classifiers on the given dataset, whereas a classifier labeled as *Class 0* indicates that the identified classifier could not be among the top performers.

For each treated dataset and subsequently evaluated pipelines, we extract the set of characteristics of the dataset (meta-features) and the characteristics of pipelines (either top performer class or low performer class). The set of meta-features vectors are then used to fit a Decision Tree prediction model for the top performing classifiers, using the meta-feature variables as predictors and the classifiers labels as targets of the meta-model. We use the decision tree to build the prediction model because of its good interpretability and the better ability to understand the coefficients of feature importance of the model [6]. For the current Meta-Model, we have mainly focused on the optimization of the recall of *Class 1* as the classifier has the potential to be among the best performing classifiers. Therefore, we were bound to consider different levels of the decision tree model hyperparameters configurations. For instance, the following configuration provided the best meta-model result (among all the considered configuration):

'class_weight': (1: 1, 0: 0.7), 'criterion': 'gini'.

Evaluation results and discussion

In the following sections, we describe the empirical study of the performance achieved by the experiments with AMLBID on various manufacturing datasets. Following the eventual experimental configuration, we demonstrate the ability of AMLBID to effectively search the generated enormous hyperparameters space to find the optimal algorithms and hyperparameters with low computational complexity. Finally, it leads us to a comparative evaluation of the performance of AMLBID to those of the currently available the state-of-the-art (i.e. the TPOT [8] and Autosklearn [10]) ML pipelines generation tools.

Experimental configuration

To ensure meaningful comparison, we benchmark on a highly varied selection of 30 datasets that cover both binary and multiclass classification problems from the perspectives of different industry 4.0 levels. These data are gathered broadly from two major sources :

- OpenML AutoML benchmark: Datasets curated to serve as representative benchmark for AutoML frameworks [43]. These datasets span various binary and multiclass classification problems and exhibit substantial heterogeneity in sample size and dimensionality.
- State-of-the-art papers: Datasets collected from research papers dealing mainly with industry 4.0 related problems using machine learning solutions.

These 30 fresh datasets were not previously exploited by any learning method during the offline phase in our framework. These are introduced to AMLBID to evaluate the pipeline recommendations (Table 5).

Table 5 List (sample) of datasets used in the evaluation

Dataset	Num Class.	Num Inst.	Task
[44]	4	959	Failure risk analysis
[45]	3	2000	Chatter prediction
[46]	2	61000	RUL prediction
APSFailure	2	60000	APS system failure prediction
Higgs	2	110000	Predictive maintenance
CustSat	2	76020	Customer satisfaction

Table 6 Performance of AutoML systems on the 30-benchmark datasets

System	>	<
AMLBID	**19**	**2**
TPOT	6	5
Auto-sklearn(V)	2	17
Auto-sklearn(E)	3	6

We count how many times the system performs better (>) or worse (<)

The best performances among all AutoML frameworks are highlighted in bold

Evaluation method

The recommended ML pipelines were trained on the benchmark datasets. Subsequently, the performances of AMLBID are compared to those of the TPOT and Auto-sklearn frameworks and also to the results of the related research papers (that served as the source of original data). For TPOT, we used the default settings (i.e. generate and evaluate 120 pipeline for each dataset). While for Auto-sklearn, we compared AMLBID with two versions, as the Auto-sklearn has a "Vanilla" version that produce a single optimal pipline (Auto-sklearn(V)) and the other version that create a set of 50 best pipelines (Auto-sklearn(E)).

It is important to note that the both baseline frameworks evaluate each generated pipeline by running that on the given dataset, whereas AMLBID immediately produces, at an imperceptible computational cost, a list of potential top pipelines configurations using its meta-knowledge base.

Experimental results

We refer to the Table 6 to consult the comparative evaluation of the results obtained by exposing the 30 datasets to the AMLBID, TPOT, and both versions of Auto-sklearn. It can be observed that AMLBID performances are comparatively better than those of the baseline even though any pipeline on the dataset was not executed, prior to the recommendation. We can also observe and position the results obtained by AMLBID as more accurate than the results obtained by the TPOT, Auto-sklearn, and the related research papers on the same datasets in Table 7.

As shown in Table 7 and illustrated in Fig. 2, the results of some ML solutions, oriented from manufacturing industry, can be improved simply through the use of better models and related hyperparameters configuration. It can be useful to note that

Table 7 Performance of AMLBID and the baseline tools on the benchmark datasets

Dataset	AMLBID	TPOT	Auto-sklearn(V)	Auto-sklearn(E)	Original paper result
[44]	**0.9374**	0.9120	0.8215	0.9283	0.85
[45]	**0.9706**	0.9517	0.9632	0.9356	0.95
[46]	**0.9941**	0.9907	0.9782	0.99	0.9895
[47]	0.9205	**0.9991**	0.9357	0.6863	0.9984
[48]	0.8971	0.6711	0.908	**0.9723**	0.9677
[49]	**0.9706**	0.7767	0.678	0.9843	0.9278
[50]	**0.8967**	0.8899	0.6783	0.7952	0.884
[51]	**0.8748**	0.7826	0.6702	0.7727	0.8659
Wafer-ds	**0.8571**	0.7312	0.8033	0.8953	–
HTRU	0.8880	0.8415	**0.9027**	0.6591	–
Cnae-9	**0.9671**	0.8803	0.7922	0.8365	–
Gas_Sens	0.9739	**0.9843**	0.9256	0.9468	–
Covertype	**0.8344**	0.7307	0.7890	0.6521	–
Kc1	**0.8793**	0.7097	0.7697	0.8552	–
jannis	0.6719	**0.7229**	0.6171	0.6845	–
MiniBooNE	**0.9645**	0.9423	0.8343	0.8903	–
KDDCup	**0.9740**	0.8934	0.9331	0.95	-
segment	**0.9735**	0.9681	0.9337	0.9542	–
Higgs	0.713	0.726	0.7135	**0.729**	–
Credi-g	**0.7921**	0.7188	0.5739	0.6121	–
shuttle	0.9649	**0.9905**	0.8429	0.9362	–
APS Failure	0.9910	**0.9933**	0.9716	0.984	–
nomao	**0.9708**	0.9570	0.6995	0.7987	–
CustSat	**85.59**	0.8276	0.8072	0.8290	–
kr-vs-kp	**0.9976**	0.9209	0.6532	0.7593	–
car	0.9754	**0.9999**	0.8549	0.9462	–
albert	**0.8759**	0.8005	0.8288	0.7981	–
airlines	0.6982	0.6758	**0.7094**	0.5927	–
Numerai28.6	**0.5207**	0.4229	0.4836	0.4433	–

The best performances among all AutoML frameworks are highlighted in bold

evaluating only the top-1 recommended pipeline makes AMLBID more efficient than Auto-sklearn and TPOT.

In most of the state-of-the-art AutoML systems, one of the shortcomings is their computational complexity. Often, they require huge time and resources budget on non-conventional datasets. On the contrary, AMLBID has the advantage of the O(1) computational complexity, generating the recommendation in significantly the negligible amount of time. This argument can be further testified by the proven results shown in Table 8, which presents the performance of AMLBID, TPOT and Autosklearn in terms of execution time on the same machine for the benchmarked datasets.

The landscape performance difference of AMLBID is accomplished because the other AutoML systems consume massive time to train the multiple algorithms with various configurations on the same dataset to produce the recommendation. As they require to train the ML model from scratch for the fresh datasets prior to generate the list of recommendation configurations. Whilst, as stated in the previous section, the AMLBID

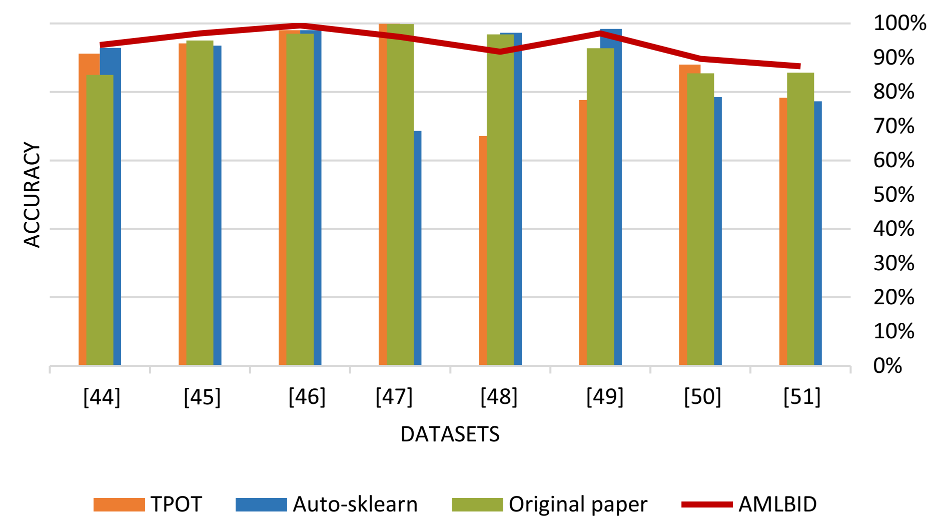

Fig. 2 The cumulative gains chart of AMLBID and the baseline AutoML tools over the state-of-the-art datasets

Table 8 The run-time (in HH:MM:SS format) of the AMLBID, Autosklearn and TPOT tools on the benchmark datasets

Dataset	Dataset size	AMLBID	Autosklearn	TPOT
[44]	959	**00:00:05**	01:23:47	00:08:14
[45]	2000	**00:00:12**	01:49:21	00:13:57
[46]	61000	**00:05:29**	04:19:05	03:42:09
[47]	274627	**00:11:43**	08:19:37	06:09:51
[48]	5000	**00:01:27**	02:31:07	01:38:36
[49]	1567	**00:00:53**	01:33:45	00:19:47
[50]	5388	**00:00:57**	01:56:50	00:55:51
[51]	1567	**00:00:33**	00:58:50	00:21:12
Wafer-ds	7306	**00:02:17**	03:44:26	01:42:21
HTRU	54641	**00:06:59**	03:42:09	02:57:11
vehicle	8463	**00:02:28**	02:12:40	01:45:40
Cnae-9	63260	**00:05:47**	04:07:39	03:24:52
Gas_Sens	4188	**00:01:14**	02:47:20	00:42:36
Covertype	25524	**00:03:04**	01:28:31	01:36:14
Kc1	2108	**00:00:38**	04:19:26	04:51:02
jannis	8641	**00:01:41**	02:31:07	01:41:51
MiniBooNE	52147	**00:04:23**	03:59:56	02:11:01
KDDCup	49402	**00:05:06**	03:47:20	02:37:38
segment	2310	**00:00:25**	01:15:45	00:33:02
Higgs	110000	**00:06:16**	07:37:55	05:43:24
Credi-g	30000	**00:04:39**	02:03:34	05:33:03
shuttle	57999	**00:05:48**	05:15:45	04:26:03
APS Failure	60000	**00:05:39**	03:58:39	05:23:35
nomao	31772	**00:04:08**	03:01:15	02:49:36
CustSat	76020	**00:06:06**	05:07:03	04:09:36
kr-vs-kp	3196	**00:00:54**	01:17:19	00:22:44
car	1728	**00:00:38**	01:38:30	00:40:07
albert	43824	**00:06:27**	04:09:17	03:01:03
airlines	5473	**00:01:40**	02:18:27	00:57:52
Numerai28.6	6574	**00:03:22**	02:07:39	01:16:17

The best performances among all AutoML frameworks are highlighted in bold meta-knowledge base is equipped with more than 4 millions of evaluated pipelines hence, it is capable to generate the recommendation by comparative search of meta-features with most similar existing dataset. Furthermore, with each iteration of fresh dataset, the meta-knowledge base of AMLBID is further enriched with evolutionary training. The confidentiality of the fresh dataset is respected by the fact that the knowledge base of AMLBID consists of the meta-features of dataset and not the data.

We can evidently conclude, from any of these obtained results, that AMLBID is significantly more accurate than all the baseline AutoML frameworks.

AMLBID software package

AMLBID has been implemented as a Python-package. The AMLBID API is subjected to automate the algorithms selection and related hyperparameters tuning for supervised machine learning through the proposed meta-learning based framework. The tool broadly has the `AMLBID_Recommender` module for recommending and building highly tuned ML pipelines according to the desired predictive metric.

Script 1 summarizes the interactions required to use AMLBID to recommend a ML pipeline. Subsequently, it attributes a score to the chosen pipelines and export the optimal one to a dynamically stored Python (`.py`) file.

```
 1  from AMLBID.recommender import AMLBID_Recommender
 2  from AMLBID.loader import *
 3
 4  #Load dataset
 5  Data,X_train,Y_train,X_test,Y_test=load_data("Dataset.csv")
 6
 7  #Generate the optimal configuration
 8  model=AMLBID_Recommender.recommend(Data, metric="Accuracy",
 9                                            mode="Recommender"
10  model.fit(X_train, Y_train)
11
12  print(model.score(X_test, Y_test))
13
14  #Export configuration's corresponding Python code
15  model.export('Recommended_pipeline.py')
```

Script 1: Illustrative code example of recommendation module.

As shown on line 5, the root directory of the dataset to be loaded is specified. The `recommend` function (as shown on line 8) initiates the meta-learning process to identify the top performing recommendations (ML algorithm with related hyperparameters configuration) based on the given performance criteria. Next, the recommended pipeline is evaluated on the provided test-set samples (as shown on line 12). Once this code completes its execution, the `export` function (as shown on line 15) is used to generate and export, `Recommended_pipeline.py` (shown in script 2), which is the corresponding Python implementation of the recommended ML pipeline.

```python
import numpy as np
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report
from sklearn.model_selection import train_test_split

data = pd.read_csv("Dataset.csv")

X = data.drop('class', axis=1)
Y = data['class']

X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
      test_size=0.3, random_state=42)

model= DecisionTreeClassifier(criterion='entropy',
                              max_features=0.5672564,
                              min_samples_leaf=5,
                              min_samples_split=20)

model.fit(X_train, Y_train)

Y_pred = model.predict(X_test)
score = model.score(X_test, Y_test)

print(classification_report(Y_test, Y_pred))
print(' Pipeline test accuracy:  %.3f' % score)
```

Script 2: Generated python file.

Despite being in its early stages, the AMLBID package has been already appealed by the community as can be witnessed on the PyPI's[7] Python packages index. Feedback from the community is highly positive, and several new applications have been proposed in addition to multiple research requests.

Conclusion and future work

To enhance the production quality and improve the manufacturing industry, new techniques and technologies are being developed consistently. The machine learning techniques has been promising for the interests of the fourth industrial revolution. However, the customised knowledge required to use ML during the training process has been among the major obstacles to incorporate and advance ML models in manufacturing industrial domains. The dependence on manual crafting of ML models to produce desired performance makes it a difficult task despite the proven potential of ML models to improve the production.

The current work has been focused on the design, implementation, and evaluation of an automated ML system for manufacturing industry. AMLBID, in this context, is a novel MtL-based tool that address the problematic of automated selection and configuration of ML algorithms. The proposed tool uses a recommendation engine that incorporates a meta-knowledge base maintained by the previous and ongoing

[7] https://pypi.org/project/AMLBID/.

recommendation results in manufacturing domain. The AMLBID explicitly train meta-models which are capable of identifying effective pipelines by exploring the interactions between datasets and pipelines topology without performing expensive computational analysis. In this regard, we presumably prevail the major limitations of AutoML-based systems which have been the computational complexity and excessive run-time performance losses. The presented comprehensive empirical evaluations reveal that AMLBID is significantly more accurate than popular AutoML frameworks. In the future, we are planning to extend the proposed framework in several directions. We have set the short-term goal to expand AMLBID to include datasets and models for regression problems. Furthermore, we are also aiming to extend the analysis to include the classifiers of distributed ML libraries since we encounter the analysis of industrial big data. In addition, further analysis and Meta-Models can also be developed.

Abbreviations
ML: Machine learning; HPs: Hyperparameters; MtL: Meta-learning; AutoML: Automated machine learning; kNN: k-Nearest Neighbors; SVM: Support Vector Machines; SGD: Stochastic Gradient Descent; RF: Random Forest; GNB: Gaussian Naive Bayes; DT: Decision Trees; LDA: Linear Discriminant Analysis.

Acknowledgements
The authors thank the University of the Littoral Cote d'Opale, School of engineering's and business' sciences and technics (HESTIM) Morocco, and CNRST Morocco for the financial support, and the CALCULCO computing platform, supported by SCoSI/ULCO (Service COmmun du Système d'Information de l'Universitédu Littoral Côte d'Opale) for the computational facilities.

Author contributions
MB, AA and MG developed the research idea and conceptualized the research theme. MG designed, implemented the algorithms and wrote the manuscript. He also conducted the research experiments in order to validate the approach. MG, AA and MB defined the research questions and verified the results. MG, AA, MB, MH, AL and GB participated in the general research guidance and scientific queries for the research workflow control and management. All authors read and approved the final manuscript.

Funding
Not applicable.

Availability of data and materials
Not applicable. For any collaboration, please contact the authors.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Author details
[1]UR 4491, LISIC, Laboratoire d'Informatique Signal et Image de la Cote d'Opale, Univ. Littoral Côte d'Opale, 62100 Calais, France. [2]CCPS Laboratory, ENSAM, University of Hassan II, Casablanca, Morocco. [3]Study and Research Center for Engineering and Management (CERIM), HESTIM, Casablanca, Morocco.

Received: 21 January 2022 Accepted: 11 April 2022
Published online: 29 April 2022

References

1. Razali NAM, Shamsaimon N, Ishak KK, Ramli S, Amran MFM, Sukardi S. Gap, techniques and evaluation: traffic flow prediction using machine learning and deep learning. J Big Data. 2021. https://doi.org/10.1186/s40537-021-00542-7.

2. Lehmann C, Huber LG, Horisberger T, Scheiba G, Sima AC, Stockinger K. Big data architecture for intelligent maintenance: a focus on query processing and machine learning algorithms. J Big Data. 2020. https://doi.org/10.1186/s40537-020-00340-7.
3. Ed-daoudy A, Maalmi K. A new internet of things architecture for real-time prediction of various diseases using machine learning on big data environment. J Big Data. 2019. https://doi.org/10.1186/s40537-019-0271-7.
4. Ruiz-Sarmiento J-R, Monroy J, Moreno F-A, Galindo C, Bonelo J-M, Gonzalez-Jimenez J. A predictive model for the maintenance of industrial machinery in the context of industry 4.0. Eng Appl AI. 2020. https://doi.org/10.1016/j.engappai.2019.103289.
5. Wolpert DH, Macready WG. No free lunch theorems for optimization. IEEE Trans Evol Comput. 1997;1(1):67–82. https://doi.org/10.1109/4235.585893.
6. Garouani M, Ahmad A, Bouneffa M, Lewandowski A, Bourguin G, Hamlich M. Towards the automation of industrial data science: a meta-learning based approach. In: Proceedings of the 23rd international conference on enterprise information systems. 2021. p. 709–16. https://doi.org/10.5220/0010457107090716.
7. Hutter F, Kotthoff L, Vanschoren J. Automated machine learning. 1st ed. Cham: Springer; 2019. https://doi.org/10.1007/978-3-030-05318-5.
8. Olson RS, Moore JH. TPOT: a tree-based pipeline optimization tool for automating machine learning. In: Automated machine learning: methods, systems, challenges. Cham: Springer; 2019. p. 151–60. https://doi.org/10.1007/978-3-030-05318-5_8.
9. Kotthoff L, Thornton C, Hoos HH, Hutter F, Leyton-Brown K. Auto-WEKA: automatic model selection and hyperparameter optimization in WEKA. In: Automated machine learning: methods, systems, challenges. Cham: Springer; 2019. p. 81–95. https://doi.org/10.1007/978-3-030-05318-5_4.
10. Feurer M, Klein A, Eggensperger K, Springenberg JT, Blum M, Hutter F. Auto-sklearn: efficient and robust automated machine learning. In: Automated machine learning: methods, systems, challenges. Cham: Springer; 2019. p. 113–34. https://doi.org/10.1007/978-3-030-05318-5_6.
11. Kotu V, Deshpande B. Predictive analytics and data mining: concepts and practice with RapidMiner. Cambridge: Morgan Kaufmann; 2015.
12. BigML. https://bigml.com/. Accessed 01 Nov 2022.
13. DataRobot. https://www.datarobot.com/. Accessed 01 Nov 2022.
14. Lechevalier D, Narayanan A, Rachuri S, Foufou S. A methodology for the semi-automatic generation of analytical models in manufacturing. Comput Ind. 2018;95:54–67. https://doi.org/10.1016/j.compind.2017.12.005.
15. Villanueva Zacarias AG, Reimann P, Mitschang B. A framework to guide the selection and configuration of machine-learning-based data analytics solutions in manufacturing. Procedia CIRP. 2018. https://doi.org/10.1016/j.procir.2018.03.215.
16. Xu Z, Dang Y, Munro P. Knowledge-driven intelligent quality problem-solving system in the automotive industry. Adv Eng Inform. 2018;38:441–57. https://doi.org/10.1016/j.aei.2018.08.013.
17. Asif K, Zhang L, Derrible S, Indacochea JE, Ozevin D, Ziebart B. Machine learning model to predict welding quality using air-coupled acoustic emission and weld inputs. J Intell Manuf. 2020. https://doi.org/10.1007/s10845-020-01667-x.
18. Cuartas M, Ruiz E, Ferreño D, Setién J, Arroyo V, Gutiérrez-Solana F. Machine learning algorithms for the prediction of non-metallic inclusions in steel wires for tire reinforcement. J Intell Manuf. 2021;32(6):1739–51. https://doi.org/10.1007/s10845-020-01623-9.
19. Wang X-B, Luo L, Tang L, Yang Z-X. Automatic representation and detection of fault bearings in in-wheel motors under variable load conditions. AEI. 2021;49: 101321. https://doi.org/10.1016/j.aei.2021.101321.
20. Gao Y, Yu D. Intelligent fault diagnosis for rolling bearings based on graph shift regularization with directed graphs. Adv Eng Inform. 2021;47:101253. https://doi.org/10.1016/j.aei.2021.101253.
21. Zhou C, Chase JG, Rodgers GW. Degradation evaluation of lateral story stiffness using HLA-based deep learning networks. Adv Eng Inform. 2019;39:259–68. https://doi.org/10.1016/j.aei.2019.01.007.
22. Medina R, Jean Carlo M, Pablo L, Diego C, Sánchez R-V, Mariela C. Gear and bearing fault classification under different load and speed by using Poincaré plot features and SVM. J Intell Manuf. 2020. https://doi.org/10.1007/s10845-020-01712-9.
23. Usuga Cadavid JP, Lamouri S, Grabot B, Pellerin R, Fortin A. Machine learning applied in production planning and control: a state-of-the-art in the era of industry 4.0. J Intell Manuf. 2020;31(6):1531–58. https://doi.org/10.1007/s10845-019-01531-7.
24. Carbonneau R, Laframboise K, Vahidov R. Application of machine learning techniques for supply chain demand forecasting. Eur J Oper Res. 2008;184:1140–54. https://doi.org/10.1016/j.ejor.2006.12.004.
25. Wu Q. Product demand forecasts using wavelet kernel support vector machine and particle swarm optimization in manufacture system. JCAM. 2010;233(10):2481–91. https://doi.org/10.1016/j.cam.2009.10.030.
26. Wuest T, Weimer D, Irgens C, Thoben K-D. Machine learning in manufacturing: advantages, challenges, and applications. Prod Manuf Res. 2016;4(1):23–45. https://doi.org/10.1080/21693277.2016.1192517.
27. Garouani M, Ahmad A, Bouneffa M, Hamlich M, Bourguin G, Lewandowski A. Towards big industrial data mining through explainable automated machine learning. Int J Adv Manuf Technol. 2022. https://doi.org/10.1007/s00170-022-08761-9.
28. Drori I, Krishnamurthy Y, Rampin R, Lourenço R, Ono JP, Cho K, Silva C, Freire J. AlphaD3M machine learning pipeline synthesis. arXiv:2111.02508.
29. Bilalli B, Abelló A, Aluja-Banet T, Munir RF, Wrembel R. PRESISTANT: data pre-processing assistant. In: Information systems in the big data era. Cham: Springer; 2018. p. 57–65. https://doi.org/10.1007/978-3-319-92901-9_6.
30. Vainshtein R, Greenstein-Messica A, Katz G, Shapira B, Rokach L. A hybrid approach for automatic model recommendation. CIKM '18. 2018. p. 1623–6. https://doi.org/10.1145/3269206.3269299.
31. Reif M, Shafait F, Goldstein M, Breuel T, Dengel A. Automatic classifier selection for non-experts. Pattern Anal Appl. 2014;17(1):83–96. https://doi.org/10.1007/s10044-012-0280-z.

32. Bisong E. Building machine learning and deep learning models on google cloud platform. Berkeley: Apress; 2019. https://doi.org/10.1007/978-1-4842-4470-8.
33. Guyon I, Sun-Hosoya L, Boullé M, Escalante HJ, Escalera S, Liu Z, Jajetic D, Ray B, Saeed M, Sebag M, Statnikov A, Tu W, Viegas E. Analysis of the AutoML challenge series 2015–2018. In: AutoML. Springer series on Challenges in Machine Learning; 2019.
34. Swearingen T, Drevo W, Cyphers B, Cuesta-Infante A, Ross A, Veeramachaneni K. Atm: a distributed, collaborative, scalable system for automated machine learning. In: 2017 IEEE international conference on Big Data (Big Data). 2017. p. 151–62. https://doi.org/10.1109/BigData.2017.8257923.
35. Olson RS, Bartley N, Urbanowicz RJ, Moore JH. Evaluation of a tree-based pipeline optimization tool for automating data science. In: Proceedings of the genetic and evolutionary computation conference 2016. GECCO '16. 2016. p. 485–92. https://doi.org/10.1145/2908812.2908918.
36. Kotthoff L, Thornton C, Hoos HH, Hutter F, Leyton-Brown K. Auto-WEKA 2.0: automatic model selection and hyper-parameter optimization in WEKA. J Mach Learn Res. 2017;18(25):1–5.
37. Ian HW, Eibe F, Mark AH. Data mining: practical machine learning tools and techniques. 4th ed. Cambridge: Morgan Kaufmann; 2017. https://doi.org/10.1016/C2015-0-02071-8.
38. Garouani M, Ahmad A, Bouneffa M, Hamlich M. AMLBID: an auto-explained automated machine learning tool for big industrial data. SoftwareX. 2022;17:100919. https://doi.org/10.1016/j.softx.2021.100919.
39. Ahmadvand H, Goudarzi M, Foroutan F. Gapprox: using gallup approach for approximation in big data processing. J Big Data. 2019. https://doi.org/10.1186/s40537-019-0185-4.
40. Al-Mansoori A, Abawajy J, Chowdhury M. Cost-aware big data stream processing in cloud environment. In: Lecture notes of the institute for computer sciences, social informatics and telecommunications engineering. Berlin: Springer; 2021. p. 120–36. https://doi.org/10.1007/978-3-030-69992-5_10.
41. Ahmadvand H, Dargahi T, Foroutan F, Okorie P, Esposito F. Big data processing at the edge with data skew aware resource allocation. In: 2021 IEEE conference on network function virtualization and software defined networks (NFV-SDN). 2021. https://doi.org/10.1109/nfv-sdn53031.2021.9665051.
42. Wolf H, Lorenz R, Kraus M, Feuerriegel S, Netland TRH. Bringing advanced analytics to manufacturing: a systematic mapping. In: Advances in production management systems. Production management for the factory of the future. 2019. p. 333–40. https://doi.org/10.1007/978-3-030-30000-5_42.
43. Gijsbers P, LeDell E, Thomas J, Poirier S, Bischl B, Vanschoren J. An open source AutoML benchmark. 2019. arXiv:1907.00909.
44. Mazumder RK, Salman AM, Li Y. Failure risk analysis of pipelines using data-driven machine learning algorithms. Struct Saf. 2021;89:102047. https://doi.org/10.1016/j.strusafe.2020.102047.
45. Saravanamurugan S, Thiyagu S, Sakthivel NR, Nair BB. Chatter prediction in boring process using machine learning technique. IJMR. 2017;12(4):405. https://doi.org/10.1504/IJMR.2017.088399.
46. Benkedjouh T, Medjaher K, Zerhouni N, Rechak S. Health assessment and life prediction of cutting tools based on support vector regression. J Intell Manuf. 2015;26(2):213–23. https://doi.org/10.1007/s10845-013-0774-6.
47. Anton SDD, Sinha S, Dieter Schotten H. Anomaly-based intrusion detection in industrial data with SVM and random forests. In: 2019 international conference on software, telecommunications and computer networks (SoftCOM). 2019. p. 1–6. https://doi.org/10.23919/SOFTCOM.2019.8903672.
48. Deng H, Diao Y, Wu W, Zhang J, Ma M, Zhong X. A high-speed d-CART online fault diagnosis algorithm for rotor systems. Appl Intell. 2019;50(1):29–41. https://doi.org/10.1007/s10489-019-01516-2.
49. Kim JK, Han YS, Lee JS. Particle swarm optimization-deep belief network-based rare class prediction model for highly class imbalance problem. Concurr Comput Pract Exp. 2017. https://doi.org/10.1002/cpe.4128.
50. Imoto K, Nakai T, Ike T, Haruki K, Sato Y. A CNN-based transfer learning method for defect classification in semiconductor manufacturing. IEEE Trans Semiconduct Manuf. 2019;32(4):455–9. https://doi.org/10.1109/tsm.2019.2941752.
51. Kim JK, Cho KC, Lee JS, Han YS. Feature selection techniques for improving rare class classification in semiconductor manufacturing process. In: Lecture notes of the institute for computer sciences, social informatics and telecommunications engineering. Berlin: Springer; 2017. p. 40–7. https://doi.org/10.1007/978-3-319-58967-1_5.

Publisher's Note