Towards big industrial data mining through explainable automated machine learning

Moncef Garouani1,2,3 · Adeel Ahmad1 · Mourad Bounenna1 · Mohamed Hamlich2 · Gregory Bourguin1 · Arnaud Lewandowski1

Received: 28 July 2021 / Accepted: 14 January 2022 / Published online: 10 February 2022
© The Author(s), under exclusive licence to Springer-Verlag London Ltd., part of Springer Nature 2022

Abstract
Industrial systems resources are capable of producing large amount of data. These data are often in heterogeneous formats and distributed, yet they provide means to mine the information which can allow the deployment of intelligent management tools for production activities. For this purpose, it is necessary to be able to implement knowledge extraction and prediction processes using Artificial Intelligence (AI) models, but the selection and configuration of intended AI models tend to be increasingly complex for a non-expert user. In this paper, we present an approach and a software platform that may allow industrial actors, who are usually not familiar with AI, to select and configure algorithms optimally adapted to their needs. Hence, the approach is essentially based on automated machine learning. The resulting platform effectively enables a better choice among the combination of AI algorithms and hyper-parameters configurations. It also makes it possible to provide features of explainability of the resulting algorithms and models, thus increasing the acceptability of these models in practicing community of the users. The proposed approach has been applied in the field of predictive maintenance. Current tests are based on the analysis of more than 360 databases from the subjected field.

Keywords Machine learning · AutoML · Explainable AI · Data analysis · Decision-support systems · Industry 4.0

1 Introduction
Data driven decision-making may be defined by the set of practices aiming to make decisions based on data analysis rather than on intuitive insights [1]. The business entities that have deployed data-driven decision-making activities have been observed as more profitable and productive compared to the traditional ones [2]. Nowadays, decision making tools are mainly based on results of the current AI research works [3]. The success of AI-based tools is mainly due to the advances in machine learning approaches [4]. This is particularly stimulated by the availability of large datasets concerning various real-world features [3] and also through the increase of the computational gains which are generally attributed to the powerful GPU cards [5].

The manufacturing area is one of those generating huge amounts of data gathered by means of Cyber Physical System (CPS) devices. The availability of such data combined with the knowledge of manufacturing experts may be an opportunity to build AI-based processes and models providing high value insights and assets for decision makers. Nevertheless, building such processes and models requires AI and data science skills and expertise that are not always available in the manufacturing area workbenches and laboratories.

The work shown in this paper aims to bridge the gap between AI expertise and manufacturing experts. We believe that automated machine learning (AutoML) [6] is one of the most powerful approaches that can effectively deal with this problem. Instead of searching appropriate Machine Learning (ML) algorithms and manually tuning the hyper-parameters in a virtually infinite space, the AutoML leads to automatically and iteratively configure the appropriate hyper-parameters of multiple machine learning algorithms in a virtually infinite...
space, in order to optimize these parameters for a predefined search space.

The use of AutoML approaches has gained significant attention in research community. A plethora of systems for AutoML, such as Auto-sklearn [7], TPOT [8], Auto-WEKA [9], ATM [10], and Google Cloud AutoML [11] have been developed in recent years.

The interest of building complex AI models that are able to achieve unprecedented performance levels has been gradually replaced by a growing concern for alternative design factors leading to an improved usability of the resulting tools. Indeed, in a manifold of application areas, complex AI models become of limited practical utility [12]. The major reason lies on the fact that AI models are often designed to focus the performance factors, thus leaving aside other important and even sometimes the crucial aspects such as confidence, transparency, fairness or accountability. The absence of explanation for predicted performing factors make the AI models usually black boxes, which only allows the prominent exhibition of input and output parameters but conceal the visibility of inherent associations among them. It is more preferably desired to avoid such lack of transparency in real-life applications such that in industrial manufacturing processes. Since, these applications may imply critical decision choices, it is favorable to have some justifications of individual predictions which are perceived through an AI algorithm, more particularly, in an automated environment. Therefore acceptance of, and the trust in, an AutoML system is highly dependent on the transparency of the recommendations.

Because of the lack of transparency in AutoML systems as Decision Support Systems (DSS), users tend to question the validity of automatic results, such that: did the AutoML run long enough? Did the AutoML miss some suitable models? Did the AutoML sufficiently explore the search space? Did the recommended configuration over or under fit?, etc. Such queries may cause reluctance for users to apply the results of AutoML in more critical situations [13]. Meanwhile, when AutoML provides unsatisfactory results, users are unable to reason and thus cannot improve the obtained results. They may only increase the computational budget (e.g., the run-time) as much as possible, which can result as barriers of the AutoML effectiveness.

It is therefore a preliminary objective of the current work to make the outcome from such well-performing AutoML systems transparent, interpretable and self-explainable. This shall make AutoML support systems more reliable and operational through a set of different visual summary levels of the provided models and configurations. It may render the AutoML system more transparent and controllable, hence increasing its acceptance.

In the current work, we attempt a transparent and auto-explainable AutoML system for recommending the most adequate ML configuration for a given problem and explain the rationale traceability behind a recommendation. It may further allow to analyze the predictive results in an interpretable and reliable manner. In the proposed approach, the end users can explore the AutoML process at different levels, such as described in the following:

- The AutoML-oriented level (i.e. exploring the AutoML process from recommendation to refinement).
- The Data-oriented level (i.e. exploring data properties through different visualization levels).
- The Model-oriented level (i.e. exploring the models provided by the AutoML system (e.g. model performance, what-if-analysis, decision path, etc.)).

The system consists of two integrated modules: the Automated Machine Learning tool for Big Industrial Data (AMLBID) module and the Automated Machine Learning explainer (AMLExplainer). However, the later, AML-Explainer module is not system or algorithm specific; it is inter-operable with a variety of AutoML frameworks. The main contributions of this work are summarized as follows:

1. We present AMLBID, a premier transparent, interpretable and auto-explainable meta-learning [14] based AutoML tool that identifies the optimal or near-optimal ML configuration for a given problem.
2. Provide an assisted traceability of the reasoning behind the AutoML recommendation generation process.
3. Develop a module that can explain the predictions of any recommendation through linked visual summary and/or textual information.
4. Provide a multi-level interactive visualization tool that facilitate the model operation and performance inspection to address the “trusting the model”.
5. Provide a reliable guidance, when AutoML returns unsatisfying results in order to improve the expected performances by assessing the importance of an algorithm hyperparameters.

The rest of the paper is organized as follows: Sect. 2 discusses the closely related works in respect of ML-based data analytics solutions and the need for transparency to gain trust in AI models. Section 3 briefly describes the different types of explanations and also their respective information content with their use in practice. Section 4 provides an overview of the proposed framework and discusses how these components collaborate to achieve the pursued goals. Finally, the contents of this paper are concluded in Sect. 5 along with outlines of future perspectives.
2 Related works

The available literature testifies a considerable progress in respect of automated machine learning methodologies and their application in multi-disciplinary areas. We have been attentively studying some of the most relevant works, particularly the recommendation-based decision support systems in manufacturing industry. We particularly considered the works dealing with transparency and explainability of automated machine learning and those related to big industrial data mining.

Through the advances on high-tech sensing and the widespread use of applications such as electronic manufacturing records, mobile sensors, and Industrial Internet of Things (IIoT) tools, manufacturing data are being accumulated at an exponentially growing rate annually. According to a recent research report [15], the global big data market size will grow from USD 138.9 billions in 2020 to USD 229.4 billion by 2025 (increase from Petabytes to Exabytes). Machine learning is a key technology to transform large manufacturing data sets, or “big industrial data”, into actionable knowledge.

In this section, we observe the limitations of available research works on explainable AI/AutoML systems as DSS that primarily motivate the work described in this paper. The current literature, in this regard can be generally summarized in form of an overlapped overview of three major research areas, as described in the following subsections.

2.1 Challenges in selecting and configuring machine learning algorithms

The machine learning is widely used in many industrial applications across different levels, including processes, machines, shop floors, and supply chain levels. For instance, machine learning models can be used to control product quality [16], to monitor the condition of tools by tracking the evolution of their state [17], or to monitor the health of machines by predicting the time of occurrences of machine failures and also to estimate the criticality of these failures [18]. However, despite its countless benefits and advances, building a machine learning pipeline is still a challenging task, partly because of the difficulty in manually selecting an effective combination of an algorithm and hyperparameters values for a given task or problem.

Owing to the development of open source ML packages and the active research in the ML field, there are dozens of machine learning algorithms, where each machine learning algorithm has two types of model parameters: (1) ordinary parameters that are automatically optimized or learned during the model training phase; (2) and hyperparameters (categorical and continuous) that are typically set by the user manually before the training of the model (as shown in Table 1).

To achieve the desired performance for a particular problem, users typically try a set of models and configurations based on their understanding of the algorithms and their observation of the data since there is no algorithm that performs well on all possible problems (i.e., No Free Lunch [19]). Then, based on the feedback about how the learning tools performed, the practitioner may adjust the configuration to verify if the performance can be improved. Such a trial-and-error process terminates once a desired performance is achieved or the computational budget runs out (as shown in Fig. 1).

2.2 Automated machine learning

The Automated Machine Learning or AutoML [20] field is among the rapidly emerging sub-fields of ML that attempts to address the theoretical and algorithmic challenges in order to fully automate the ML process. It addresses also the development and deployment of systems in this regard. AutoML has two main goals: (1) democratizing the application of ML to non-experts of data analysis by providing them with “off the shelf” solutions, and (2) enabling the knowledge practitioners to save time and effort.

Over the last few years, a plethora of AutoML systems have been developed, providing partial or complete ML automation, such as Auto-sklearn [7], TPOT [8], AutoWEKA [9], ATM [10], as well as commercial systems such as Google AutoML [11], RapidMiner [21], DarwinAI [22], and DataRobot [23]. These tools range from automatic data preprocessing [24, 25], automatic feature engineering [26, 27] to automatic model selection [20, 28] and automatic hyperparameters tuning [29, 30]. Some approaches attempt to automatically and simultaneously choose a learning algorithm and optimize its hyperparameters. These approaches are also known as Combined Algorithm Selection and Hyperparameters optimization problem (CASH) [7–9, 30–33]. Table 2 shows a comparison among some of the most popular AutoML tools, in terms of training framework,

---

Table 1 Configuration space of some classification algorithms

| ML Algorithm          | Number of Ordinary Parameters | Number of Hyperparameters |
|-----------------------|-------------------------------|---------------------------|
| SVM                   | 2                             | 5                         |
| Decision Tree         | 1                             | 3                         |
| Random Forest         | 2                             | 4                         |
| Logistic Regression   | 4                             | 6                         |
supported ML tasks, automatic features engineering, user interface and process transparency.

While all of these tools provide partial or complete ML process automation, each one works differently and targets different dataset structure, platform, algorithm, or end user. Similarly, they have simultaneously unmatched advantages and disadvantages. For instance, Auto-Sklearn is embedded in Python, however, it only operates on structured data. Auto-WEKA supports Weka ML algorithms with the provision of a graphical user interface (GUI), but it is limited to statistical algorithms. RapidMiner provides features engineering capability but requires expert guidance. Likewise, Google AutoML supports most types of datasets and algorithms but the dedicated service is commercial and it is only available as cloud-based.

The interest in developing complex AI models, despite their revolutionary characteristics and capabilities of achieving unprecedented levels of performance has been progressively perishing. The loss of sustainability is mainly concerned with alternative design factors particularly to make such models more usable in practice [3]. These systems emphasize on useful assistance but their usability is greatly limited to provide detailed analysis about the recommended configurations and the insightful working of the black-box models [13]. The reason lies on the fact that AI models are often designed with performance as their only design goal, thus ignoring other important matters such as privacy awareness, confidence, transparency, and accountability makes them usually untrustworthy black-boxes [13, 34].

| Tool               | Training Framework | ML task         | Automatic features engineering | User interface | Transparent |
|--------------------|--------------------|-----------------|-------------------------------|----------------|-------------|
|                    |                    |                 | Categorical data processing    | Missing values | Self-explainable |
| Google AutoML*     | TensorFlow, SparkML| Classification & Regression | NO | YES | YES | YES |
| DarwinAI*          | TensorFlow        | Classification   | YES | YES | YES | YES |
| RapidMiner*        | Scikit-learn, TensorFlow, Spark | Classification & Regression | YES | YES | YES | YES |
| AutoSklearn[7]     | Scikit-learn      | Classification & Regression | NO | YES | NO | NO |
| TPOT [8]           | Scikit-learn      | Classification & Regression | YES | NO | NO | NO |
| AutoWeka[9]        | Weka              | Classification & Regression | NO | YES | YES | NO |
| AutoKeras [31]     | Keras             | Classification   | NO | NO | NO | NO |
| SmartML[33]        | RWeka             | Classification   | YES | NO | YES | NO |

(*) : commercialized tools
2.3 The need for transparency to trust in AI and in AutoML

Black-box AI systems have been used in various areas. Their implication in critical domains, like in power consumption forecasting or supply chain management to analyze the brands trends or consumer sentiments; usually have less focus to consider the quality features such as transparency and explainability rather considering more importantly the system’s overall performance. However, even if these systems fail, e.g., the Quality Control system is mostly not able to detect the failure, the Equipment Failure Prevention system are less expected to identify the exact cause of failure and generally produces false or inaccurate predictions. The consequences are rather undermining. In industrial critical applications, the situations are different where the lack of transparency of ML techniques can be a disqualifying factor, if not limited. Specifically, a single wrong decision can be highly risked to put in danger the entire production line (e.g., failure of a critical unit) and can cause significant financial deprivations (e.g., product conformity). It is therefore, relying on an incomprehensible black-box data-driven system would not be the best option. The lack of transparency is among the most relevant reasons to question the adoption of AI models in manufacturing industry. The stakeholders are more cautious than doing so in the consumer entertainment, or e-commerce industries.

Explaining the reasoning behind one’s decisions or actions is an important part of human interactions in the social dimension [35]. As the explanations help to build trust in human-to-human relationships, similarly, these should also be part of human-to-machine interactions [3]. In this work, we investigate the contributions and feasibility of a process designed to make such powerful DSS transparent, interpretable and self-explainable to foster trust, both in situations where the AI system has a supportive role (e.g., production planning) and in those where it provides directions and decision-making (e.g., Quality Control, predictive maintenance or autonomous driving). In the former cases, explanations provide extra information, which help the human in the loop to gain an overall view of the situation or the problem at hand in order to take decisions. It is similar to an expert who has to provide a detailed report explaining his/her findings, a supportive AI system should explain the decisions in detail instead of providing only a prediction or a decision.

3 Explainable AI

Explainable AI (XAI) [12] refers to artificial intelligence technologies that can provide human-understandable explanations for their output or actions [36]. End users, by nature, may wonder about the reasoning behind how and why algorithms make or arrive to decisions [34]. As the complexity of the AI algorithms and systems grows, they are viewed as “black-boxes” [32]. Increasing complexity can result in the lack of transparency that hampers understanding the reasoning of these systems, which negatively affects the users trustiness.

Model explainability can be divided into two categories: global explainability and local explainability. Global explainability means the users can understand the model directly from its overall structure. Local explainability just consider a specific input and it tries to find out why the model makes a certain decision.

The development of methods for explaining, visualizing and interpreting machine learning models has recently gained increasing attention under the Explainable AI (XAI) area [12, 13, 34, 36]. In the recent years, the advancements in XAI are grown rapidly but there are still broader gaps to generalize XAI approaches. The current major XAI methodologies are only applicable to specific types of data and models. Such specificities mostly require the pre-configuration of input parameters that are not easily coded by non-experts. In contrast, our proposed system intends to support the analysis and inspection of all machine learning classification models without any data type dependency, neither even having to write any line of code. A variety of XAI methods characteristics in terms of data explanations level, data and model dependency, and pre-configuration requirements are highlighted in Table 3.

Table 3: Properties of XAI state of the art tools. Level is the interpretability coverage: local or global. Dependency specifies necessary inputs.

| XAI method          | Level | Dependency | Require pre-configuration |
|---------------------|-------|------------|---------------------------|
|                     | Local | Global     | Data | Model |                    |
| LIME [13]           | •     | o          | ?    | o     | •                   |
| ANCHORS [37]        | •     | o          | •    | o     | •                   |
| Node-Link Vis [38]  | •     | •          | o    | •     | •                   |
| SHAP [39]           | •     | •          | o    | o     | •                   |
| DeepTaylor [40]     | •     | o          | •    | •     | •                   |
| Occlusion [41]      | •     | o          | •    | •     | •                   |
| AMLExplainer        | •     | o          | o    | o     | o                   |

It can be useful to primarily establish a consensus of understanding on what the term explainability may refer in the context of artificial intelligence and, more specifically, in the area of machine learning. Different levels of explanations provide insights into different aspects of the model, ranging from information about the learned representations to the identification of distinct prediction strategies and the assessment of the overall model behavior. Depending on the
recipient and his or her intent, it may be advantageous to focus on one particular level of explanation.

Recent design recommendations put more focus on the importance of intuitive interfaces, along with a clean and concise presentation, among the explanation facilities, and easy user interactions [42]. In order to make AI systems as decision support systems accessible and effortless for both machine learning experts and neophytes, system builders and designers should present not only the final model prediction or recommendation coming out of the system, but also the pipeline steps and decisions made in each of those steps along with the prediction or decision generation process. In this argument, more clarity and transparency for users are expected from the AutoML system. To accommodate the user needs of transparency and trust, a few recent works have proposed prototypes design for increasing AI systems transparency [42–44]. However, most of these systems fall short in providing an overview of the AutoML process of how and why a recommended pipeline configuration was generated, and their interfaces are complex and not suitable for manufacturing routine and needs. Furthermore, the incorporated explanation facilities are often insufficient and/or not tailored to the industrial domain.

Our proposed system aims to provide guidance to solve particular problems (Fig. 2). Given a dataset, the tool automatically recommends the most adequate ML configurations and allow users to easily observe and analyze these models through an interactive multiple views module that explain the inner working of any machine learning classifier in an interpretable and faithful manner. The goal is manifold: (1) facilitate the models working and performance inspection through linked visual summaries and textual information (2) provide a visual summary of all evidence items and their relevance for the computation result, and (3) present a guided investigation of the reasoning behind the recommendation generation.

4 The conceptual framework

Given a predictive modeling problem for an industrial application, it is often difficult to build an accurate machine learning-based predictive model that is easy to develop and to be interpreted by non-ML experts. The key idea for our transparent and explainable automated machine learning vision is to separate recommendations from explanations by using two modules simultaneously. The first module is used for making the recommendation of the most adequate ML configuration for a problem at hand and aims to maximize the requested predictive performance metric (e.g. Accuracy, Precision, Recall). The second module is used for providing the rationale behind the recommended configuration as well as automatically explaining the inner workings of the model.

The following section describes the design and implementation choices of the proposed tool, a complete, transparent and self-explainable AutoML system. As it is shown by Fig. 3. For the recommender module (AMLBID), AutoML is performed when a new dataset is presented, and a list of candidate pipelines is provided based on the given task. The dataset characteristics, the output of AutoML and the list of candidate pipelines are supplied to the explanatory module in order to generate an interactive dashboard to better understand the provided results. It also helps the end-user to diagnose the performance of the generated pipelines and explore the possibilities of performance refinement.

4.1 The recommender module

The AMLBID module is a Meta-learning [14] based system for automating the algorithm selection and tuning problem using a recommendation system that is bootstrapped with a collaborative meta-knowledge base. This knowledge-base, derived from a large set of experiments conducted on 360 real-world datasets from different manufacturing levels generating more than 3 millions of different ML configurations (pipelines). Each pipeline consists of a choice of a machine learning model and its hyperparameters configuration. By exploring the interactions between datasets and pipelines topology, the system is able to identify effective pipelines without performing computationally expensive analysis.

Building a meta-learning-based system to deal with the algorithms selection and configuration problem requires a meta-knowledge base for the learning process. This involves collecting datasets, choosing machine learning algorithms, extracting meta-features (datasets and pipelines characteristics), and determining the performance of the algorithm configurations according to different evaluation measures (e.g. Accuracy, precision, Recall). Though, when a new
problem is presented to the system, the meta-features are extracted, and a recommendation mechanism that makes use of the meta-knowledge base provides the ranking of the pipeline(s) (algorithms and configurations) for the unseen problem according to the desired performance measure.

AMLBID consists of two main phases: the learning phase and the inferring one. During the learning phase, we evaluate different classification algorithms, analyze multiple datasets (to extract meta-features), and train a ranking meta-model. During the inference phase, the meta-model generated in the training phase is used to produce a ranked list of promising classification pipelines for a new dataset and a classification performance metric.

4.1.1 The learning phase

During the learning phase, we evaluate different classification algorithms with multiple hyperparameters configurations on a large collection of various datasets. Then, we generate meta-features that are used to train a meta-model able to recommend promising classification pipelines for a given dataset and performance metric. The entire training phase is illustrated in Fig. 4.

The datasets We conduct the experiments on 360 real-world manufacturing classification datasets which are collected...
from the popular UCI1, OpenML2, Kaggle3 and KEEL4 repositories, among other real-world scenarios. These datasets cover various tasks with respect to their size, the number of attributes, their composition and class imbalance.

It is worth noting that the used datasets cover a broad range of application areas. They include, among others, the process level studies, machine related problems and supply chain level (as shown in Table 4). We have not performed any preprocessing operation on the datasets to avoid any potential bias or impact on the classifiers performances and in order to ensure the fairness for the performance comparisons.

### The meta-features

The meta-learning core paradigm is to relate the performance of learning algorithms and configurations to data characteristics (Meta-features). Meta-features are common characteristics of several problems, and their aim is to identify structural similarities and differences among problems. These characteristics can be divided into three categories [33]:

**Simple** Based on general measures, such as the number of instances, attributes and classes, dataset dimensionality. They are designed to some extent to measure the complexity of the underlying problem.

**Statistical** Based on statistical measures obtained from the dataset attributes, such as means, standard deviation, class entropy, and correlations, etc.

**Landmark** That characterize the extent of datasets when basic machine learning algorithms (with default configuration) are performed on them. The used landmark characteristics in our system include performance of the linear discriminant analysis (LDA), Gaussian Naive Bayes (GB), Decision Trees (DT), Gaussian Naive Bayes (GNB) and the K-Nearest Neighbor (KNN) landmarks.

1. https://archive.ics.uci.edu/
2. https://www.openml.org/
3. https://www.kaggle.com/
4. https://sci2s.ugr.es/keel/

### The pipelines generation

To build the Meta-knowledge base, we used 08 classifiers from the popular Python-based machine learning library, Scikit-learn. These classifiers are AdaBoost, Support Vector Classifier (SVC), Extra Trees, Gradient Boosting, Decision Tree, Logistic Regression, Random Forest, and Stochastic Gradient Descent (SGD) classifiers. Detailed description of the algorithms and their tuned hyperparameters are described on the Tables 11–17 in the Appendix.

For each run of a classifier $C$ over a dataset $D$, we generated 1000 different combinations of their hyperparameters configurations. This process resulted in an average of 8000 pipelines per dataset. In particular, for each classifier, we have generated a list of all possible and reasonable combinations where we conducted, for each dataset, a random search among them [45].

During the training phase, we used a fivefold stratified cross-validation strategy to construct our meta-datasets. As a result, our knowledge base consists of more than 3 millions evaluated classification pipelines. It is noted that due to the different number of algorithms hyperparameters, not every algorithm had the same number of configurations/evaluations. The knowledge-base is continuously improved by running more tasks, making AMLIBID smarter by achieving more experience, based on the growing knowledge-base.

### Measures

As part of our core idea, we aim to recommend high-performing ML pipelines for a given combination of datasets and evaluation measure. The point that most of state-of-the-art systems do not take into account, the proposed system supports various classification performance measures to evaluate the performance of the ML pipelines (ML algorithms and related hyperparameters configuration). Table 5 shows supported measures details.

#### 4.1.2 The recommending phase

The **recommending phase** is initiated when a new dataset to be analyzed occurs. At this point, the user selects a predictive analytic metric to be used for the analysis (e.g. Accuracy, Recall, F1 score), and then the system automatically recommends a set of machine learning algorithms and their related hyperparameters configuration to be applied, such that the predictive performance is the first-rate. To do so, the system first, extracts the dataset characteristics (meta-features). Then, the extracted meta-features are fed to the meta-model to provide the candidate pipelines. Finally, the suggestion engine, according to the meta-knowledge base, ranks the pipelines in respect to the provided analytic metric. The recommending process is shown in Fig. 5.

---

**Table 4** Statistics about the used datasets according to related tasks

| Task                  | Number of Datasets | Average of Attr. | Inst. | Classes |
|-----------------------|--------------------|------------------|-------|---------|
| Process level         | 78                 | 29               | 30529 | 3       |
| Machine level         | 248                | 53               | 13942 | 2       |
| supply chain level    | 74                 | 17               | 21726 | 2       |

---

---
Meta-model After having generated a meta-dataset with all the necessary metadata, the goal is to build a predictive meta-model that can learn the complex relationship between a task meta-features and the utility of specific ML pipelines to recommend the most useful ML algorithm(s) configuration given the meta-features $M$ of a new task $t_{\text{new}}$.

Formally, each task $t_j \in \mathcal{T}$ is described by a vector $F(t_j) = (m_{j1}, \ldots, m_{jK})$ of $K$ meta-features $m_{j,k} \in F$, the set of all known meta-features. This can be used to define the task similarity measure based on, for instance, the Euclidean distance between $m(t_{\text{new}})$ and $m(t_j)$, so that we can transfer information from the most similar tasks to the new task $t_{\text{new}}$.

The distance measured between meta-features of $t_{\text{new}}$ and $t_j$ is given by Eq. (1):

$$d(m(t_{\text{new}}), m(t_j)) = \sqrt{\sum_{i=1}^{K} (m_{\text{new},i} - m_{t,j,i})^2}$$  \hspace{1cm} (1)

One of the aims of our work is to produce an enriched meta-model able to recommend the top-performing classification configuration(s) for a combination of an unseen dataset and classification evaluation measure. For this purpose, two state of the art learning algorithms were used to produce meta-models able to predict the most appropriate pipelines for the dataset at hand: Random Forest (RF) and k-Nearest Neighbor

| Measures   | Description                                      | Importance                                      |
|------------|--------------------------------------------------|--------------------------------------------------|
| Precision  | Precision considered as a measure of exactness or quality. | Precision is used to retrieve fraction of instances that are relevant. |
| Recall     | Recall is a measure of completeness or quantity.  | Recall is used to retrieve fraction of relevant instances that are retrieved. |
| Accuracy   | The accuracy is the proportion of the total number of predictions that were correct. Accuracy is related to the degree of bias in the measurements | Accuracy is used to represent the correct answer or percentage of accurate classification. |
| F1 score   | F1 score or F-measure is defined as the harmonic mean of precision and recall. Commonly used as a single metric to evaluate the classifier performance. | A value closer to one implies that a better combined precision and recall is achieved by the classifier |

Table 5 Supported classification measures

![Fig. 5 Recommending phase workflow](image)
Thus, when the meta-learning system is applied to a new dataset, the meta-model returns a ranking of the most suitable classification algorithms and configurations, based on its meta-feature values.

Ranking using KNN classifier is a commonly used strategy to obtain the top-K rankings. When a new dataset is presented to the meta-learning system, the KNN identifies the closest neighbors of the candidate dataset in the meta-knowledge base, using the Euclidean distance measure (Eq. (1)). Based on this measure, a vector $d = [d_1, d_2, \ldots, d_k]$ containing the dissimilarity among all characteristics (meta-features) of datasets is built and a weighted average of each individual neighbor’s is used for forecasting the optimal pipeline configuration based on the relevant measure.

While for the Random Forest meta-model, we produce for each supported classification evaluation measure $E$ a large labeled training set using the following process:

1. For each combination of $d \in D$ and $A^{(i)} \in A$, where $D$ is the 360-learning datasets, $A^{(i)}$ a learning algorithm configuration from the 3 millions evaluated configurations, we retrieve the set of all best predictive results $R(E_i, d, A^{(i)})$ for each evaluation metric $E_i$ (e.g., accuracy, F1-score, recall and precision).

2. For each $d \in D$ we designate the learner algorithm configuration $A^{(i)}$ as Class 1 (top performer algorithm configuration for the dataset) if its best predictive results for the dataset are greater than or equal to the highest performance achieved by all other algorithm configurations. Otherwise we label the $A^{(i)}$ for the dataset as Class 0 (low performer algorithm configuration).

3. For each combination of $d \in D$ and $A^{(i)} \in A$ we generate a joint set $F = \{ F_d \cup F_{A^{(i)}} \}$, where:

   - $F_d$: the dataset’s meta-features generated in the learning step.
   - $F_{A^{(i)}}$: a discrete feature describing the learning configuration $A^{(i)}$.

4. The joined meta-features vectors $F$ are used to fit the RF meta-model for the top performing algorithms configurations, using the meta-features variables as predictors and the learner’s labels as targets of the meta-model. For our Meta-Model, we have been mainly interested for optimizing the prediction recall of Class 1 (the classifier has the potential to be among the best performing classifiers). Therefore, we had to consider different levels of the decision tree model hyperparameters configuration where the configuration: $\{\text{class_weight} : \{1 : 1, 0.7\}, \text{criterion} : \text{gini, max_features} : \text{None}\}$ provided the best meta-model result.

The main functionality of the meta-model can be formally defined as follows: given a set of learning algorithms space $A = \{A^{(1)}, \ldots, A^{(n)}\}$ where $i$ is the $i$-th hyperparameters configuration of $A$, a dataset $D$ divided into disjoint training $D_{\text{train}}$ and validation $D_{\text{validation}}$ sets, and an evaluation measure $E$, the goal is to identify the ML algorithm (s) $A^{(i)}$, where $A^{(i)} \in A$ and $A^{(i)}$ is a tuned version of $A^{(i)}$ that minimizes or maximizes the $E$ on $D$.

$$A^{(i)} \in \text{Argmin}_{A^{(i)} \in A} E(A^{(i)}, D_{\text{train}}, D_{\text{validation}})$$ (2)

Figure 6 presents Random Forest and KNN meta-models performances on suggesting the best predictive pipeline configuration. The KNN-based meta-model clearly performs better than the random forest classifier-based meta-learner according to the accuracy metric.

### 4.1.3 The evaluation of robustness

In this evaluation, we investigate the performance that can be achieved by using the proposed recommending module on various manufacturing related problems. We evaluate its ability to predict the ML algorithms with related hyperparameters configurations that shall provide the best result of the analysis. We benchmark on a highly varied selection of 30 datasets covering binary and multi-class classification problems from different industry 4.0 levels with a sample size from 1000 to 100000 instance (this is a common sample size for general real-world datasets). These 30 fresh datasets were not previously exploited by any learning method during the offline phase in our framework. Table 6 shows characteristics of a sample of the evaluation datasets in terms of number of classes, instances and covered tasks.

The proposed system uses the meta-model to predict all the pipelines in the meta-knowledge base with respect to the analyzed dataset and then returns its top-ranked pipelines according to the provided performance criteria. The
The top-ranked pipelines are then fitted on the datasets that was split into train and test sets using a 70% / 30% ratio. The results of this evaluation were used to compare the performances of AMLBID to those of the TPOT and Auto-sklearn state-of-the-art frameworks. It is important to highlight, that while the majority of state-of-the-art frameworks evaluate a set of pipelines by running them on the given dataset before the recommendation step, which demands considerable computational budget that is not always available. They also take a huge amount of time in the majority of cases, which makes them impractical solutions in real-world problems as in the industrial contexts. AMLBID immediately produces a ranked list of potential top pipelines configurations using its meta-model and meta-knowledge base at an imperceptible computational cost (in term of time and computational resources). The evaluation results are presented in Table 7.

As shown in Table 7 and summarized in Table 8, it is clear that the performances of AMLBID are comparable and even better than those of the baselines even though it does not run any pipeline on the dataset prior to the recommendation.

As state of the art systems support only the predictive accuracy as the performance measure of the recommended configuration, further robustness comparison on different performance measures such Recall, F1 score and Precision could not be done. Whereas in some concrete cases, Recall or Precision may be more important and informative than the predictive accuracy. Therefore, the proposed system is the first AutoML system to support different predictive performance measures (Precision, Recall, Accuracy and F1 score).

Beyond their black box nature, among the major shortcomings of AutoML solutions is their computational complexity, which necessitates a huge budget of time and resources. On the contrary, AMLBID has the advantage of the $O(1)$ computational complexity, consequently it generates the recommendation in negligible amount of time. Table 9 presents the running time of AMLBID, TPOT and Autosklearn on the benchmarked datasets.

Noting that the rather “long” time taken by some massive datasets for AMLBID relates to the calculations made for extracting the dataset’s characteristics (Meta-features).

### 4.2 The explainer module

AMLExplainer is implemented as a client-server tool integrated with the recommender module. The server coordinates as an AutoML support system. As the client, the visual interface provides graphical interaction with AutoML results and maps the summary data for visualization through a set of different visual summary levels of the recommended models. We structure the process of explainability in the proposed XAI framework during the following three phases:

#### Understanding phase

This phase is the entry point of the proposed explainability workflow. For a ML expert, this step offers necessary information to fit and deploy the recommended model(s). For a model user (supposedly, a non-ML expert), it explains the recommended models and their functionalities by providing visual representations, descriptions and external information about the recommended model(s). In the current version of the prototype system, we implement this phase as the integration of information cards of the properties.

### Table 6

| Dataset                  | Num. of Class | Num. of Inst | Task               |
|--------------------------|---------------|--------------|--------------------|
| Mazumder et al. [46]     | 4             | 959          | Failure risk analysis |
| Saravanamurugan et al. [47] | 3             | 2000         | Chatter prediction  |
| Benkedjouh et al. [48]   | 2             | 61000        | RUL prediction     |
| APS Failure              | 2             | 60000        | APS failure prediction |
| Higgs                    | 2             | 110000       | Predictive maintenance |
| CustSat                  | 2             | 76020        | Customer Satisfaction |

### Table 7

| Dataset                  | AMLBID   | TPOT     | Auto-sklearn |
|--------------------------|----------|----------|--------------|
| Mazumder et al. [46]     | 93.74    | 91.20    | 92.83        |
| Saravanamurugan et al. [47] | 97.06    | 95.17    | 93.56        |
| Benkedjouh et al. [48]   | 99.41    | 99.07    | 99           |
| APS Failure              | 99.10    | **99.33** | 98.4         |
| Higgs                    | 72.6     | 72.6     | **72.9**     |
| CustSat                  | **85.59**| 82.76    | 82.90        |

The best performances among all AutoML frameworks are highlighted in bold.

### Table 8

| System      | Number of Datasets with Top Performance |
|-------------|----------------------------------------|
| Autosklearn | 3 (10%)                                |
| TPOT        | 6 (20%)                                |
| AMLBID      | 21 (70%)                               |
summary and classification statistics of the recommended model(s) along with a report regarding the input data.

Diagnosis phase In the proposed framework, we define the diagnosis phase as the most important part of the XAI workflow. It enables end users (experts or novices), to visually explore the insight of the recommended ML models and thus understand the reasoning behind the output of the model. It offers a decision support tool, that helps end users examine the attribution value for each input feature with respect to the model predictions through the what-if analysis and the decision path.

Refinement phase This phase shows the interaction recommendations. These are based on the recommended model architecture, findings from previous stages (meta-Knowledge base), and general heuristics (ANOVA [49]) to improve the predictive performance of the recommended model(s).

AMLExplainer users are allowed to explore the models provided by the AutoML process at four main levels of detail (i.e. AutoML Overview, Recommendation-level View, What-if analysis-level View, and Refinement-level View). Meanwhile, AMLExplainer provide end users with a guidance, when AutoML returns unsatisfying results, to improve the predictive performances. Thence increases the transparency, controllability, and the acceptance of AutoML.

The workflow of the proposed auto-explanatory AutoML system consists of two main components:

- The AutoML component, which shows the high-level of the AutoML process from recommendations to refinements.
- The recommended configuration component, that allows users to inspect the recommended model’s inner working and decision’s generation process (includes the Recommendation-level and What-if analysis-level views).

The AutoML overview The AutoML overview level (Fig. 7) summarizes high-level information of the AutoML process. Users will be able to compare and choose between the top K recommended configurations. They can focus their analysis on the top model configuration on the next level view, which highlights the corresponding algorithm in the detail views.

Table 9 The running time of AMLBID and the baseline AutoML frameworks on the benchmark datasets. The running time format is HH:MM:SS

| Dataset           | Ds size | AMLBID    | TPOT       | Autosklearn |
|-------------------|---------|-----------|------------|-------------|
| Mazumder et al.   | 95      | 00:00:05  | 00:08:14   | 01:23:47    |
| Saravanamurugan et al. | 2000   | 00:00:12  | 00:13:57   | 01:49:21    |
| Benkedjouh et al. | 61000   | 00:05:29  | 03:42:09   | 04:19:05    |
| APS Failure       | 60000   | 00:05:39  | 05:23:35   | 03:58:39    |
| Higgs             | 110000  | 00:06:16  | 05:43:24   | 07:37:55    |
| CustSat           | 76020   | 00:04:06  | 04:09:36   | 05:07:03    |

The best performances among all AutoML frameworks are highlighted in bold.

The International Journal of Advanced Manufacturing Technology (2022) 120:1169–1188

Fig. 7 AutoML overview
The recommendation-level view

The recommendation-level view enables users to inspect recommendations with respect to performance distribution. As shown in (Fig. 8), a detailed explanation about the top performed recommendation is generated through multiple granularity levels, such as statistics about the configuration performances (Fig. 8(A)), and a tree-based explanation of the conducted predictions (Fig. 8(B)).

By providing intelligible explanations about the process and reasoning behind an individual prediction, as illustrated in Fig. 8, it is clear that the decision-maker whether a manufacturing engineer or a machine learning practitioner is much better positioned to make decisions since He / She usually have prior knowledge about the data and the application domain, which can use to trust in and accept or reject a prediction if the reasoning behind it is well explained.

The what-if analysis-level view

Figure 9 is designed to investigate the machine learning models. It enables understanding models by enabling end users to investigate attribution values for individual input features in relation to model predictions. Explaining the inner working of the model helps to gain an understanding of what the model does and does not do. This is important so that they can gain an intuition for when the model is likely missing information and may have to be overruled. Therefore explore scenarios, test, and evaluate / validate business assumptions, and gain intuition for modification.

The refinement-level view

Figure 10 shows the correlation between performances and hyperparameters of a recommended algorithm. To accomplish that, we takes as input performances data gathered with different hyperparameter settings of the algorithm (from the recommender module’s knowledge-base), fits a random forest to capture the relationship between hyperparameters and performances, and then we apply functional Analysis of variance (ANOVA) to assess how important each of the hyperparameters and each low-order interaction of hyperparameters is to performance. Guided by this in-depth analysis, end users have a guidance, when AutoML returns unsatisfying results, to improve the predictive performances.

The tool documentation and a detailed list of features with an illustrative example is available in the Github repository in form of a Python-package5.

4.3 Evaluation of user explanations

The following sections describe the evaluation methodology of the proposed auto-explainable AutoML tool. We globally draw the insights from the feedback that we have received from the various target users.

4.3.1 Demonstration test case: application to manufacturing quality prediction

The proposed system is designed for the machine learning-based predictive modeling problems. The major goal of the work has been to show the feasibility to achieve maximum possible performance for a specific predictive modeling problem, and automatically explain the results

---

5 https://github.com/LeMGarouani/AMLBID
for any machine learning predictive model. We evaluate the intuitiveness and the usability of the automatic explanation method on a Manufacturing Quality Prediction use case (real-life environment). The data contains 187,156 historical 1-year records of a production unit. Among these records, 74.39% are diagnosed as compliant products.

4.3.2 User interview

Participants and apparatus To evaluate the proposed white-box AutoML system as a decision support system, we conducted a semi-structured qualitative user study with two different groups of target users. These groups range from ML novices to experts (53% male and 47% female who are aged between 24 and 38 years with an average age of \( \mu = 26.78 \) years). Among the ML-users, 48% were the participants with particular knowledge in the big industrial data analysis. While, among the ML-experts, 52% were the participants with experience in developing ML models for their domain problems. All of these participants have the experience in machine learning or data analysis, but none of them had prior experience with AutoML. The evaluation studies are conducted on a set of dedicated computers equipped with an Intel Core i5-2400 3.10GHz - 8Go RAM DDR4.

Tasks and procedure The evaluation study began with a tutorial session, in which the tasks and the usage of the self-explainable AutoML system were introduced to the participants. Participants were asked to complete the Post-Study
System Usability Questionnaire (PSSUQ), third version [50], when performing the explanatory module on the three tasks understand, diagnose, and refine.

The PSSUQ-3 is a 16 item measure (as shown in the questionnaire sheet Fig. 12 of Appendix). The questionnaire consists of an overall satisfaction scale (the mean value of items 1 through 16) and three subscales. System usefulness subscale assesses the ease of learning and use of the system (i.e. the mean value of items 1 through 6). The information quality subscale evaluates the feedback provided by the system to the user (i.e. the mean value of items 7 through 12). Finally, the interface quality subscale quantifies the familiarity of the user with the system, as whether the system has met the expected functionality (i.e. the mean value of items 13 through 16). For all the scales, the rating range was between 1 and 7; the lower the score, the higher the satisfaction with the tool.

The results of the usability questionnaire are summarized in Table 10 and Fig. 11. The PSSUQ overall and subscale scores were extremely positive, with an overall total mean score of 1.53 (standard deviation 0.71) and a range from strongly agree to neutral (1 to 5). In this context, the mean system usefulness, information quality, and interface quality subscale scores were 1.74, 1.09, and 1.22 respectively.

As shown in Fig. 11, most of the participants agreed that the auto-explainable AutoML DSS is easy to learn and use. Among them, 80% of the participants strongly agreed that they are confident in their recommended model(s). We also conducted semi-structured post-study interviews to gather more detailed feedback from participants. The interviews reflect the difference between the initial expectation and the experience during the pair analytic regarding the workflow of the system.

We collected the participants feedback about the AutoML module as a black-box decision support system assisting the experts to choose and configure ML models for their problems initially and afterwards with the entire system (recommendation module and the explanation module). Based on their feedback, we summarize two main appreciations of the proposed tool:

- AutoML can help stakeholders (neophyte as well as experts) to improve the applications of machine learning algorithms. AutoML enables quick experimentation with a large number of models and configurations, whose results could provide useful knowledge to ML researchers and domain practitioners. On the test set, the recommended machine learning predictive model configuration achieved an accuracy of 97.81% while their configuration based on their understanding of the algorithms and their observation of the data achieved a predictive accuracy of 91.42%. These findings can inform the users about
the importance of hyperparameters tuning for ML algorithms. The participants highlight the fact that being able to match prior knowledge about machine learning to the visualizations produced by AMLExplainer creates confidence in the underlying AutoML process and increases the likelihood of adopting AutoML.

The participants appreciated the human-machine interaction introduced in AMLExplainer. They observed such interaction could improve an AutoML process and enhance user experience and make such powerful black boxes trustworthy. One of the experts commented: “Users with more domain knowledge, such as myself, are usually critical of automated methods and like to be in control. I do not like getting a score back and hearing trust me”.

Overall, the feedback on the system remains positive. In addition, the users provided several suggestions for complementary features. For the understanding and diagnosis tasks, in addition to the provided explanation levels, users wanted to gain insight into the underlying data. Such exploratory data analysis feature [51] is an integral part for any knowledge discovery process. For example, a data profiling level could review the dataset characteristics and quality and show it to the user. For results reporting, the feedback is mostly unified. All participants liked the code export function of the recommended ML pipeline and they are aspirant to use that for the communication of their results. Furthermore, in the perspectives of the code export, participants came up with several suggestions for enhancing this feature, such that possibility to store/export the overall explanation levels as PDF report file.

For the refinement task, there are mixed feedback and expectations. Most of the participants are optimistic and they suggest additional ways to interactively refine the recommended model(s). Rather than providing static guidance content, some individual non-ML experts ask further guidance to select the appropriate refinements. Moreover, the ideas to enhance its functionality include the propositions of code fragments, providing building blocks, or even scaling it up to a click-to-refine functionality. In the future, the current system should be extended with the suggested refinement methods and additional guidance to select the appropriate refinements.

Our documentation of the real-world evaluation case illustrates how to overcome the transparency problem of AutoML systems as decision support systems, for instance, the absence of human interaction and analysis of the inner working and reasoning of such tools. This could extend the use-of and trust-in the intelligent AutoML systems to areas where they are so far neglected due to their insistence on comprehensible models. Separating the automatic selection and configuration of machine learning algorithms from model explanation is another benefit of expert and intelligent AutoML DSS.

5 Conclusion

There has been significant progress in democratizing the application of ML to non-experts of data analysis by providing them with “off the shelf” solutions. However, these powerful support systems fail to provide detailed instructions about the recommended configurations and the inner workings of these models, thence making them less trustworthy highly performant black-boxes. In this work, we present a novel transparent and self-explained AutoML system along with an interactive visualization module that supports machine learning experts and neophytes in analyzing the automatic results of an AutoML DSS.

To our knowledge, the proposed system is the first application of the general explanation methods of AutoML systems as decision support systems. We explore several levels of explanations, ranged from individual decisions to the entire model’s recommendations and predictions. The explanations of the prediction models and what-if analysis proved to be an effective support for manufacturing related problems. A set of evaluations demonstrate the utility and usability of AMLBID in a real-world manufacturing problem. We show how powerful black-box ML systems can be made transparent and help domain experts to iteratively evaluate and update their beliefs. Based on the promising findings presented in this paper, further validation of the proposed framework in other real-world applications with a larger and more diverse group of users shall improve the visualization and presentation of explanations. At present, we are planning to expand AMLBID to support algorithms of regression, deep learning and distributed ML libraries (e.g., Spark ML [52]) since we are dealing with Big Industrial Data.

Acknowledgements The authors thank the Université du Littoral Côte d’Opale (ULCO), France, School of engineering’s and business’ sciences and technics (HESTIM), Morocco and CNRST Morocco for the partial financial support, and all the participants involved in the system evaluation for their constructive discussions and valuable suggestions.

Availability of data and materials All data generated or analyzed during this study are included in this paper.

Code availability Software code is included in the study github repository: https://github.com/LeMGarouani/AMLBID.

Declarations

Ethics approval All authors confirm that this article does not have any academic ethics issues and strictly follows the journal submission rules.

Consent to participate All authors agree to participate in the research work of this paper and publish it in the International Journal of Advanced Manufacturing Technology.
**Appendix**

See Tables 11, 12, 13, 14, 15, 16, and 17 and Fig. 12

---

**Table 11**  SVM hyperparameters tuned in the experiments

| Hyperparameter | Values                          | Description                                                                 |
|----------------|--------------------------------|-----------------------------------------------------------------------------|
| complexity (or: ‘C’) | [1e-10, 500] (log-scale)       | Soft-margin constant, controlling the trade-off between model simplicity and model fit. |
| Kernel         | {‘poly’, ‘rbf’}                  | The function of kernel is to take data as input and transform it into the required form (linear, nonlinear, polynomial, radial basis function (RBF), and sigmoid). |
| coeff0          | [0., 10]                        | Additional coefficient used by the kernel (sigmoid kernel only).             |
| gamma           | [1e-3, 1.01] (log-scale)        | Length-scale of the kernel function, determining its locality.               |
| Degree          | [2, 3]                          | Degree for the ‘poly’ kernel.                                              |

**Table 12**  Random Forest & Extra Trees Hyperparameters tuned in the experiments

| Hyperparameter | Values                          | Description                                                                 |
|----------------|--------------------------------|-----------------------------------------------------------------------------|
| bootstrap      | {true, false}                   | Whether to train on bootstrap samples or on the full train set.             |
| Max_features   | [0.1, 0.9]                      | Fraction of random features sampled per node.                               |
| Min_samples_leaf | [1, 20]                        | The minimal number of data points required in order to create a leaf.       |
| Min_samples_split | [2, 20]                      | The minimal number of data points required to split an internal node.       |
| imputation     | mean, median, mode             | Strategy for imputing missing numeric variables.                            |
| split criterion| {entropy, gini}                 | Function to determine the quality of a possible split.                     |

**Table 13**  Adaboost Hyperparameters tuned in the experiments

| Hyperparameter | Values                          | Description                                                                 |
|----------------|--------------------------------|-----------------------------------------------------------------------------|
| algorithm      | {SAMME, SAMME.R}               | Determines which boosting algorithm to use.                                 |
| N_estimators   | [50, 501]                      | Number of estimators to build.                                             |
| learning rate  | [0.01, 2.0] (log-scale)        | Learning rate shrinks the contribution of each classifier.                  |
| Max_depth      | [1, 11]                        | The maximal depth of the decision trees.                                   |

**Table 14**  Decision Trees Hyperparameters tuned in the experiments

| Hyperparameter | Values                          | Description                                                                 |
|----------------|--------------------------------|-----------------------------------------------------------------------------|
| max features   | [0.1, 0.9]                      | Number of features to consider when computing the best node split.          |
| min_samples_leaf | [1, 21]                        | The minimum number of samples required to be at a leaf node.                |
| Min_samples_split | [2, 21]                      | The minimum number of samples required to split an internal node.           |
| criterion      | {‘entropy’, ‘gini’}             | Function used to measure the quality of a split.                           |

**Table 15**  Logistic Regression Hyperparameters tuned in the experiments

| Hyperparameter | Values                          | Description                                                                 |
|----------------|--------------------------------|-----------------------------------------------------------------------------|
| C              | [1e-10, 10] (log-scale)         | Regularization strength.                                                    |
| penalty        | {‘l2’, ‘l1’}                    | Whether to use Lasso or Ridge regularization.                               |
| Fit_intercept  | True, False                     | Whether or not the intercept of the linear classifier should be computed.   |
Table 16  SGD Classifier Hyperparameters tuned in the experiments

| Hyperparameter | Values | Description |
|----------------|--------|-------------|
| loss           | ['hinge', 'perceptron', 'log', 'squared_hinge', 'modified_huber'] | Loss function to be optimized. |
| penalty        | ['l2', 'l1', 'elasticnet'] | Whether to use Lasso, Ridge, or ElasticNet regularization. |
| learning rate  | ['constant', 'optimal', 'invscaling'] | Shrinks the contribution of each successive training update. |
| fit intercept  | [True, False] | Whether or not the intercept of the linear classifier should be computed. |
| H1 ratio       | [0., 1.] | Ratio of Lasso vs. Ridge regularization to use. Only used when the 'penalty' is ElasticNet. |
| eta0           | [0., 5.] | Initial learning rate. |
| Power_t        | [0., 5.] | Exponent for inverse scaling of the learning rate. |

Table 17  Gradient Boosting Hyperparameters tuned in the experiments

| Hyperparameter | Values | Description |
|----------------|--------|-------------|
| Learning_rate  | [0.01, 1] | Shrinks the contribution of each successive decision tree in the ensemble. |
| criterion      | ['friedman_mse', 'mse'] | The function to measure the quality of a split. |
| N_estimators   | [50, 501] | Number of decision trees in the ensemble. |
| max depth      | [1, 11] | Maximum depth of the decision trees. Controls the complexity of the decision trees. |
| Min_samples_split | [2, 21] | The minimum number of samples required to split an internal node. |
| Min_samples_leaf | [1, 21] | The minimum number of samples required to be at a leaf node. |

Fig. 12  The Post-Study System Usability Questionnaire

| Strongly agree | Strongly disagree |
|----------------|-------------------|
| 1              |                   |
| 2              |                   |
| 3              |                   |
| 4              |                   |
| 5              |                   |
| 6              |                   |
| 7              |                   |
| 8              |                   |
| 9              |                   |
| 10             |                   |
| 11             |                   |
| 12             |                   |
| 13             |                   |
| 14             |                   |
| 15             |                   |
| 16             |                   |

References

1. Provost F, Fawcett T (2013) Data science and its relationship to big data and data-driven decision making. Big Data 1(1):51–59. https://doi.org/10.1089/big.2013.1508
2. Brynjolfsson E, Hitt LM, Kim HH (2011) Strength in numbers: how does data-driven decision making affect firm performance? SSRN scholarly paper ID 1819486. Social Science Research Network, Rochester, NY. https://doi.org/10.2139/ssrn.1819486
3. Samek W, Müller KR (2019) Towards explainable artificial intelligence. In: Lecture Notes in Computer Science. Springer International Publishing, Cham, pp 5–22. https://doi.org/10.1007/978-3-030-28954-6
4. LeCun Y, Bengio Y, Hinton G (2015) Deep learning. Nature 521(7553):436–444. https://doi.org/10.1038/nature14539
5. Lindholm E, Nickolls J, Oberman S, Montrym J (2008) NVIDIA Tesla: a unified graphics and computing architecture. IEEE Micro 28(2):39–55. https://doi.org/10.1109/MM.2008.31
6. Nural MV, Peng H, Miller JA (2017) Using meta-learning for model type selection in predictive big data analytics. In: 2017 IEEE International Conference on Big Data (Big Data). pp 2027–2036. https://doi.org/10.1109/BigData.2017.8258149

7. Feurer M, Klein A, Eggensperger K, Springenberg JT, Blum M, Hutter F (2015) Efficient and robust automated machine learning. In: Proceedings of the 28th International Conference on Neural Information Processing Systems, vol 2. MIT Press, Cambridge, MA, USA, NIPS’15, pp 2755–2763

8. Olson RS, Moore JH (2019) TPOT: A tree-based pipeline optimization tool for automating machine learning. In: The Springer Series on Challenges in Machine Learning. Springer International Publishing, Cham, pp 151–160. https://doi.org/10.1007/978-3-030-05318-5_8

9. Kohothoff L, Thornton C, Hoos HH, Hutter F, Leyton-Brown K (2019) Auto-WEKA: automatic model selection and hyperparameter optimization in WEKA. In: The Springer Series on Challenges in Machine Learning. Springer International Publishing, Cham, pp 81–95. https://doi.org/10.1007/978-3-030-05318-5_8

10. Sweerinen T, Drevo W, Cyphers V, Cuesta-Infante A, Ross A, Veeramachaneni K (2017) ATM: A distributed, collaborative, scalable system for automated machine learning. In: 2017 IEEE International Conference on Big Data (Big Data), pp 151–162. https://doi.org/10.1109/BigData.2017.8257923

11. Cloud AutoML custom machine learning models. Google Cloud. https://cloud.google.com/automl (visited on 12/10/2021)

12. Xu F, Uszkoreit H, Du Y, Fan W, Zhao D, Zhu J (2019) Explainable AI: a brief survey on history, research areas, approaches and challenges. In: Tang J, Kan MY, Zhao D, Li S, Zan H (eds) Natural Language Processing and Chinese Computing. Springer International Publishing, Cham, Lecture Notes in Computer Science, pp 563–574. https://doi.org/10.1007/978-3-030-32226-6_51

13. Ribeiro MT, Singh S, Guestrin C (2016) “Why should I trust you?”: Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Association for Computing Machinery. New York, NY, USA, KDD ’16, pp 1135–1144. https://doi.org/10.1145/2939672.2939778

14. Lemke C, Budka M, Gabrys B (2015) Metalearning: a survey of trends and technologies. Artif Intell Rev 44(1):117–130. https://doi.org/10.1007/s10462-013-9406-y

15. R. a. M. ltd. Big data market by component, deployment mode, organization size, business function (operations, finance, and marketing and sales), industry vertical (BFSI, manufacturing, and healthcare and life sciences), and region - global forecast to 2025

16. Cuartas M, Ruiz E, Ferreño D, Setién J, Arroyo V, Gutiérrez-Solana F (2020) Machine learning algorithms for the prediction of non-metallic inclusions in steel wires for tire reinforcement. J Intell Manuf. https://doi.org/10.1007/s10462-020-1623-9

17. Medina R, Macancela JC, Lucero P, Cabrera D, Sánchez RV, Cerrada M (2020) Gear and bearing fault classification under different load and speed by using Poincaré plot features and SVM. J Intell Manuf. https://doi.org/10.1007/s10462-020-01712-9

18. Jalali A, Heistracher C, Schindler A, Hashofeir B, Nemeth T, Glawar R, Sihn W, De Boer P (2019) Predicting time-to-failure of plasma etching equipment using machine learning. In: 2019 IEEE International Conference on Prognostics and Health Management (ICPHM). pp 1–8. https://doi.org/10.1109/ICPHM.2019.8819404

19. Wolpert D, Macready W (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82. https://doi.org/10.1109/4235.585893

20. Reif M, Shafait F, Goldstein M, Breuel T, Dengel A (2014) Automatic classifier selection for non-experts. Pattern Anal Applic 17(1):83–96. https://doi.org/10.1007/s10044-012-0280-z

21. Babicheva SA, Ries J, Lvovsky AI (2002) Quantum scissors: teleportation of single-mode optical states by means of a nonlocal single photon. Preprint at https://arxiv.org/abs/quant-ph/0208066v1

22. Darwinai. https://darwinai.com/ (visited on 12/10/2021)

23. DataRobot. https://www.datarobot.com/ (visited on 12/10/2021)

24. Bilalí B, Abelló A, Aluja-Banet T, Wrembel R (2016) Automated data pre-processing via meta-learning. In: Bellatreche L, Pastor Ó, Almendros Jiménez JM, Ait-Amour Y (eds) Data and Model Engineering, Springer International Publishing, Cham, Lecture Notes in Computer Science, pp 194–208. https://doi.org/10.1007/978-3-319-45547-1_16

25. Bilalí B, Abelló A, Aluja-Banet T, Munir RF, Wrembel R (2018) Presistant: Data pre-processing assistant. In: Mendling J, Mournaditís H (eds) Information Systems in the Big Data Era, Springer International Publishing, Cham, Lecture Notes in Business Information Processing, pp 57–65. https://doi.org/10.1007/978-3-319-92901-9_6

26. Khurana U, Samulowitz H, Turaga D (2017) Feature engineering for predictive modeling using reinforcement learning. arXiv e-prints 1709.arXiv:1709.07150

27. Nargesian F, Samulowitz H, Khurana U, Khalil EB, Turaga D (2017) Learning feature engineering for classification. pp 2529–2535

28. Vainshtein R, Greenstein-Messica A, Katz G, Shapira B, Rokach L (2018) A hybrid approach for automatic model recommendation. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management, Association for Computing Machinery. New York, NY, USA, CIKM ’18, pp 1623–1626. https://doi.org/10.1145/3232920.3239299

29. Feurer M, Springenberg JT, Hutter F (2015) Initializing Bayesian hyperparameter optimization via meta-learning. In: Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence. AAAI Press, Austin, Texas, AAAI’15, pp 1128–1135

30. Komber B, Bergstra J, Elisahm C (2014) Hyperopt-Sklearn: automatic hyperparameter configuration for Scikit-learn. Proceedings of the 13th Python in Science Conference pp 32–37. https://doi.org/10.25080/Majora-14bd3278-006

31. Jin H, Song Q, Hu X (2019) Auto-Keras: an efficient neural architecture search system. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Association for Computing Machinery. New York, NY, USA, KDD ’19, pp 1946–1956. https://doi.org/10.1145/3292500.3303648

32. Garouani M, Ahmad A, Bounefffa M, Lewandowski A, Bourguin G, Hamilch M (2021) Towards the automation of industrial data science: a meta-learning based approach. In: Proceedings of the 23rd International Conference on Enterprise Information Systems - vol. 1: ICEIS, INSTICC. SciTePress, pp 709–716. https://doi.org/10.5220/0010457107090716

33. Maher M, Sakr S (2019) SmartML: a meta learning-based framework for automated selection and hyperparameter tuning for machine learning algorithms. https://doi.org/10.5441/002/edbt.2019.54

34. Shin D, Park YJ (2019) Role of fairness, accountability, and transparency in algorithmic affordance. Comput Hum Behav 98:277–284. https://doi.org/10.1016/j.chb.2019.04.019

35. Heath RL, Bryant J (2000) Human communication theory and research: concepts, contexts, and challenges, 2nd edn. Routledge, Mahwah, N.J.

36. Gunning D, Steffik M, Choi J, Miller T, Stumpf S, Yang GZ (2019) XAI—Explainable artificial intelligence. Science Robotics 4(37). https://doi.org/10.1126/scirobots.aay7120

37. Ribeiro MT, Singh S, Guestrin C (2018) Anchors: high-precision model-agnostic explanations. In: Proceedings of the AAAI Conference on Artificial Intelligence

38. Harley AW (2015) An interactive node-link visualization of convolutional neural networks. In: Bebis G, Boyle R, Parvin B, Koracin
