Autoencoder-kNN meta-model based data characterization approach for an automated selection of AI algorithms

Moncef Garouani, Adeel Ahmad, Mourad Mohamed Bouneffa, Mohamed Hamlich

Journal of Big Data

RESEARCH

Open Access

Autoencoder-kNN meta-model based data characterization approach for an automated selection of AI algorithms

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

*Correspondence:
moncef.garouani@etu.univ-littoral.fr

[1] Univ. Littoral Côte d'Opale, UR 4491, LISIC, Laboratoire d'Informatique Signal et Image de la Cote d'Opale, 62100e 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

Abstract

The recent evolution of machine learning (ML) algorithms and the high level of expertise required to use them have fuelled the demand for non-experts solutions. The selection of an appropriate algorithm and the configuration of its hyperparameters is among the most complicated tasks while applying ML to new problems. It necessitates well awareness and knowledge of ML algorithms. The algorithm selection problem (ASP) is defined as the process of identifying the algorithm (s) that can deliver top performance for a particular problem, task, and evaluation measure. In this context, *meta-learning* is one of the approaches to achieve this objective by using prior learning experiences to assist the learning process on unseen problems and tasks. As a data-driven approach, appropriate data characterization is of vital importance for the meta-learning. Nonetheless, the recent literature witness a variety of data characterization techniques including simple, statistical and information theory based measures. However, their quality still needs to be improved. In this paper, a new Autoencoder-kNN (AeKNN) based meta-model with built-in latent features extraction is proposed. The approach is aimed to extract new characterizations of the data, with lower dimensionality but more significant and meaningful features. AeKNN internally uses a deep autoencoder as a latent features extractor from a set of existing meta-features induced from the dataset. From this new features vectors the computed distances are more significant, thus providing a way to accurately recommending top-performing pipelines for previously unseen datasets. In an application on a large-scale hyperparameters optimization task for 400 real world datasets with varying schemas as a meta-learning task, we show that AeKNN offers considerable improvements of the classical kNN as well as traditional meta-models in terms of performance.

Keywords: Algorithm selection, AutoML, Meta-learning, Meta-features, Data representation, kNN, Autoencoder

Introduction

The exponential growth of digital information has led to the widespread adoption of machine learning solutions. While ML can assist in decision-making and data analysis, human expertise is often required [1, 2]. Human interventions are required primarily as the domain experts due to the fact that they can provide unique characteristics of

the domain. It may drastically affect the performance of the algorithms. Later on, expert data-scientists are needed due to the large number of algorithms and hyperparameters configurations which otherwise make brute force infeasible search [2–4].

In such situations, algorithm selection is one of the main challenges in applying ML to a new problem. It denotes the identification of algorithm(s) (or algorithm families) that are likely to perform better on a given combination of datasets, tasks, and evaluation measures [5]. Algorithm selection and configuration (hyperparameters tuning) is a difficult process for the novice users; since the performance of an algorithm is basically a "black box" affected by multiple characteristics of the dataset. It includes instances distribution, the number of features and their composition, and the number of classes, etc [6]. The process of discovering appropriate algorithm and its optimal hyperparameters configurations can be automated in order to prevent the inherent complexities. It may also help to accelerate the testing of multiple configurations [6, 7].

Automated machine learning (AutoML) refers to decision support systems that attempt to automate all or part of the machine learning pipeline. It has been highlighted in the recent research studies [2, 8–10] that the AutoML techniques to automate the algorithm selection and configuration process (notably Auto-sklearn [5], Auto-Weka [11], and TPOT [12]) tend to be time consuming and burdensome for computational resources. It is due to the fact that they need to execute, multiple times, each candidate algorithm and configuration on the data. This fact is further emphasized for the large datasets, where even a few execution cycles may take several hours, hence making them impractical in real world scenarios [6].

An alternative approach for addressing the algorithm selection problem is meta-learning [2, 9, 13–15]. Among others, one of the aims of meta-learning is to assist the identification of the most appropriate learning algorithm (s) for a given problem by mapping datasets's characteristics to the predicted data mining performance (e.g., execution time, predictive performances, etc.). To better serve the purpose, meta-learning systems use a set of data characteristics, called meta-features, to represent and characterize data mining tasks. The meat-learning systems then identify and analyse the correlations between these attributes and the performance of learning algorithms [16]. The proper identification of data properties is essential to map tasks to learning mechanisms. Instead of executing all learning algorithms to obtain the optimal one, meta-learning is performed on the meta-data characterizing the data mining tasks to identify the optimal or near optimal learning algorithm for the given task.

As a data-driven approach, the effectiveness of meta-learning is largely dependent on the description of tasks (i.e., meta-features). In the current context, meta-learning requires meta-features that represent the primary learning tasks or datasets to transfer knowledge across them. We observe, in the available literature that several approaches in meta-learning use families of meta-features as input to quantify task similarity. It is common to compute tasks similarity as the Euclidean distance between two meta-features vectors. While these approaches have shown to be effective in simple scenarios, they exhibit clear limitations [17]. The foremost non-trivial task among the exhibited limitations is the identification and selection of relevant meta-features. Several research questions can emerge to better address these limitations such that *What criteria should we invoke to include or discard a family of meta-features?* For instance, statistical

meta-features are not always intuitive and lack expressiveness. In [18], the authors have shown how different datasets may share identical statistical properties but noticeably they have different data distribution. Ultimately, the selection of meta-features is an *ad hoc* process based on domain knowledge. It is highly desirable to develop the more predictive meta-features and select the more informative ones in order to improve the effectiveness of meta-learning [16, 19, 20].

We believe that traditional meta-features are not always capable of capturing crucial characteristics of a given task, even though some of them are very task specific [21]. This can be attributed to the fact that they only model the general characteristics of the dataset (e.g. number of instances/features, imbalance, etc.). Learning relevant meta-features can be useful to better identify the hidden relationships across tasks, to necessarily build the accurate meta-models.

A different approach that has achieved popularity in recent years invokes Deep Neural Networks (DNNs). The strength behind DNNs is their capacity to learn data characteristics from the diverse and large amount of data [22]. DNNs have had a strong impact in application areas such as image understanding and speech recognition [21, 23]. However, their use in meta-learning is still incipient and requires further investigation. The development of deep learning for features generation has been largely studied in the literature. It represents different datasets and tasks as embedding generated by trained deep networks. In [24], the authors solve different automatic speech recognition tasks through a two-step learning process. In the first step, the algorithm perform classification with DNNs, which is followed by the extraction of intrinsic features from the DNN output. In the second step, extracted features are used to improve model predictions.

Our hypothesis is that DNNs provide the means to extract intrinsic meta-features from data. In particular, autoencoder is a type of artificial neural networks offering good results due to their architecture and operations [25–28]. In this paper, we propose an instance-based algorithm, that learns latent meta-features from families of traditional ones. Its objective is to obtain meaningful and more informational meta-features. Specifically, the present work introduces AeKNN, a kNN-based algorithm with built-in latent features extraction strategy. AeKNN projects the training patterns into a lower-dimensional space, with the help of an Autoencoder (Ae). The goal is to produce new meta-features of higher quality from the initial data characteristics. In short, AeKNN combines two reference methods, k-Nearest Neighbors (kNN) and autoencoder, in order to take advantages of autoenconder in learning higher-level features. Thus, it supports kNN in performing pipelines recommendation in meta-learning paradigm.

The main contribution of this paper is the design of a novel meta-model, called as AeKNN, which combines an efficient latent features extraction mechanism (autoencoder) with a popular classification model (kNN). For the experimentation purposes, a collection of 400 real world problems and 8 ML algorithms has been used to assess the competitiveness of the proposed meta-model. It accumulate a knowledge base of more than 4 million evaluated pipelines.

The rest of the paper is organized as follows: In "Theoretical background and related works" section 2, a brief review of the closely related works is introduced, including meta-learning for algorithm selection and data characterization techniques. In "Proposed AekNN based data characterization approach" section, the proposed AeKNN

meta-model is described. "Experimental study" section describes the experiments illustrating the effectiveness of the proposed approach. Finally, "Conclusion" section provides the brief conclusion and points out the directions for the future work.

Theoretical background and related works

Meta-learning involves two basic aspects: the characterization of the learning problems (datasets), and the identification of the correlation between the optimal learning algorithm (s) and the problems characteristics. The first aspect relates to the techniques for characterizing datasets with meta-features, which constitutes the meta-data for meta-learning, whilst the second one is the learning stage at meta-level, which develops meta-models for the selection of appropriate algorithms and related hyperparameters configuration in respect of previously unseen datasets.

Meta-learning for algorithm selection

One of the main challenges in applying ML to new problems is the algorithm selection and related hyperparameters tuning, with the need to take the dataset characteristics (meta-features), type of task (classification, regression, clustering), and evaluation measure (e.g. predictive accuracy, precision, Recall, F1 score) into account. Let us consider the following contextual information to better understand the algorithm selection problems. The ASP can be formally defined as follows : given a set of learning algorithms space $\mathcal{A} = \{A^{(1)}, \ldots, A^{(i)}\}$, a dataset $\mathcal{D}$ divided into disjoint training D_{train}, and validation $D_{validation}$ sets, a task $\mathcal{T}$, and an evaluation measure $\mathcal{M}$, the goal of ASP is to identify the algorithm (s) $A^{(i)*}$ where $A^{(i)*} \in \mathcal{A}$ and $A^{(i)*}$ is a tuned version of $A^{(i)}$ that minimizes or maximizes the $\mathcal{M}$ on $\mathcal{D}$ [29]. In particular, the general statement of the algorithms selection and optimization problem is defined as :

$$A^{(i)*} \in \underset{A \in \mathcal{A}}{argmin}\ \mathcal{L}(A^{(i)}, D_{train}, D_{validation}) \tag{1}$$

where $\mathcal{L}(A^{(i)}, D_{train}, D_{validation})$ is the loss function (e.g. error rate, false positives, etc).

While it is usually necessary to rely on the human expertise (data scientists) to tackle the ASP challenges and difficulties, there has been increased interest in recent years in AutoML solutions. These solutions are proposed as decision support systems to find suitable ML pipelines for a given dataset. Existing AutoML solutions for algorithm selection are typically based on Bayesian optimization [30], deep reinforcement learning [31] evolutionary algorithms [12, 12], and budget-based evaluation [32].

While all of the above-mentioned solutions are effective, they are computationally expensive (both in terms of resources and runtime), because they require an iterative search of different models and configurations. In addition, when given a new dataset, most of the above solutions have to start the search for an optimal pipeline "from scratch" and evaluate each configuration on the given dataset before the recommendation. This limitation is particularly problematic when dealing with large datasets resulting in long processing time.

Meta-learning or learn to learn is an alternative approach for dealing with the ASP. Meta-learning paradigm aims to learn a mapping from the behavior of learning algorithms to the datasets characteristics (meta-features) that contribute to the improved

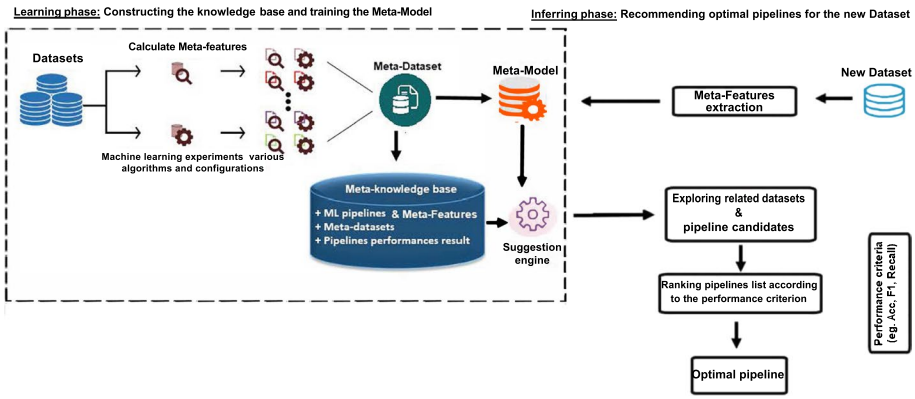

Fig. 1 The workflow of the meta-learning process

performance of one algorithm configuration over others [33]. This knowledge can be then used to better identify high-performance algorithms in order to solve the tasks on previously unseen datasets [34, 35].

As described in [36] and illustrated in Fig. 1, meta-learning frameworks consist of a collection of learning algorithms and datasets. First, the framework extracts meta-features that capture (or aspire to capture) the "essence" of a given dataset. Each learning algorithm is then applied with different related hyperparameters configurations on each dataset, and estimates the performance according to a performance measure. The extracted meta-features and the performance of the evaluated learning algorithms and configurations refer as meta-data. Then, a learning algorithm (meta-model) is trained on the meta-data to match the values of the meta-features with the most suitable algorithm (s) for each dataset. Finally, for a new dataset, the meta-learning system extracts the meta-features and uses the meta-model to recommend algorithm (s) for that dataset. Meta-data involves the extraction of significant and meaningful meta-features on the datasets or the used models [30, 34, 37, 38]. The majority of meta-features can be divided into *five families*, as shown in the next section.

Over the years, several studies have explored the application of meta-learning to various tasks such as algorithms recommendations [7–9], transfer learning [39], and assemble methods [40]. Many state-of-the-art AutoML systems use meta-learning as a way of improving their accuracy and running time [30, 41] and multiple studies describe ways to induces models from meta-knowledge as decision support systems for the ASP [6, 11–13, 37].

Some ML algorithms are often used to induce a meta-model for recommending algorithms in meta-learning. These algorithms can be an adapted version of the k-Nearest Neighbors algorithm [42]. As an example, the recommendation of the most suitable ML algorithms for a new dataset occurs by applying the kNN to the meta-features vector extracted from the new dataset.

In this study, meta-learning paradigm is used to recommend the most suitable classification algorithms for new datasets. As meta-features are crucial for the recommendation process, this work proposes a novel approach able to extract more informational features from data, allowing the recommendation meta-model to improve its performance regarding existing approaches.

Data characterization

The major task, to characterize the datasets for meta-learning, is to capture the information about learning complexity on some given dataset and identify structural similarities or differences among datasets [16]. The most early attempts to characterize datasets in order to predict the performance of classification algorithms were made by Rendell et al. [43]. We observe in the literature that broadly two main strategies are proposed subject to characterize a dataset for suggesting which algorithm is more appropriate for a specific task or dataset. Among them are the methodologies using statistical measures and a set of simplified learners. The former attempt to describe the properties of datasets using statistical and informational measures. In the later, a dataset is characterized using the training performance (e.g. accuracy) of a set of simplified learners, which became later on *Landmarking* [44].

The intuitive idea behind *Landmarking* is that the performance of classifiers is related to the intrinsic features of the problem; thus, classifiers with similar accuracy may indicate problems with similar characteristics. Characterization with the use of *Landmarkers* is known as *indirect* characterization because it is not directly related to the attributes of the problem.

The characterization of datasets using statistical and informational measures properties appeared for the first time within the framework of the STATLOG project [45]. The authors use a set of 15 characteristics, spanning from simple ones, like the number of instances and the number of attributes, to more complex ones, such as canonical correlation between the attributes and the class. This set of characteristics has been later applied in various studies for solving the ASP [9, 13, 34]. This characterization approach is later extended, it is currently known as *direct* data characterization [46] and consist of extracting simple, statistical, and information-theoretic task properties that can be straightforwardly extracted from datasets by capturing information concerning data dimensionality, distribution, and the amount of information present in the data.

Another characterization method is based on informations extracted by models built out on the problems [16]. For instance, from a decision tree model constructed over a dataset, it is possible to extract structural informations about the tree itself, such as the number of leaves, nodes, and the tree depth [47]. Similarly, in [34], the authors proposed AutoGRD, a meta-learning approach for algorithm recommendation through graphical dataset representation. First, they applied the Random Forest algorithm to create a hierarchical representation of the datasets where the vertices represent the dataset's instances and the edges indicate the existence of a sufficiently high co-occurrence score among them. Then, the GCD method [48] has been used to generate the embedding representation of the obtained graph that is fed to train an XGBoost meta-model to predict the ranking of algorithms based on their performances. However, their approach suffers from a computational complexity of O(V4) where V is the number of vertices in the analyzed graph. It is further observed that this approach is not practical for large datasets of real world problems.

Meta-features or data characteristics can be transformed to summarize the data, e.g., by reducing data dimensionality. For instance, in [49], the authors performed Principal Component Analysis (PCA) [50] to select relevant components, subsequently, a filter is used to extract the discriminating features and eliminate the redundant features.

A different approach that has achieved popularity in recent years in learning most relevant features from data involves deep autoencoder neural network. We blieve that autoencoders provide the means to extract intrinsic meta-features from traditional ones. In this process, traditional meta-features are used by the autoencoder to learn relevant features, then, the knowledge captured in the hidden layers of autoencoder is used to extract latent meta-features. Once identified and extracted, they can be used by any meta-learning algorithm.

Proposed AekNN based data characterization approach

AekNN foundations

We propose a novel approach to learn new latent meta-features by constructing new representations from traditional meta-features using a deep neural network, i.e autoencoder. An autoencoder is a type of artificial neural networks designed to learn efficient data representations (encoding) in an unsupervised manner [51]. It has a similar structure to the feedforward neural network Multi-Layer Perceptron (MLP); however, the primary difference is that the number of neurons in the output layer is *equal* to the number of inputs, whereas the autoencoder tries to generate the inputs from the learnt representation (encoding) as close as possible to its original input.

Consequently, in its simplest form, an autoencoder uses hidden layers to try to recreate the inputs. We can describe this algorithm in two parts :

1. an encoding function $Z = E(X)$ that converts X inputs to Z codings, and
2. a decoding function $X' = D(Z)$ that produces a reconstruction of the inputs X'.

The goal is to create a reduced set of codings that adequately represents X by minimizing the reconstruction error $L(X, X')$, which measures the differences between the original input data X and the consequent reconstruction X'. Formally, it can be shown as follows :

$$L(X, X') = \frac{1}{2} \sum_{i=1}^{N} \parallel x_i - x_i' \parallel_2^2 \mid i \in \{1, \ldots, N\} \tag{2}$$

The general architecture of an autoencoder is described by the number of hidden layers l_i^n and by the number of neurons per layer, where i is the index for the hidden layer and n is the total number of neurons in that layer. Each layer contains a learnt latent representation of the input data. Encoded hidden layer in the middle of the autoencoder, often called the bottleneck layer, comprises the final learnt latent features, where each latent variable is a representation of the original input in an abstract space. The number of latent variables is user defined by controlling the number of neurons in that layer. By training an autoencoder on the traditional meta-features space, we can learn a new representation (latent meta-features). The resulting deep neural network serves as a features extractor where the learnt latent space Z is extracted from the middle hidden layer. This process is highlighted in Figure 2.

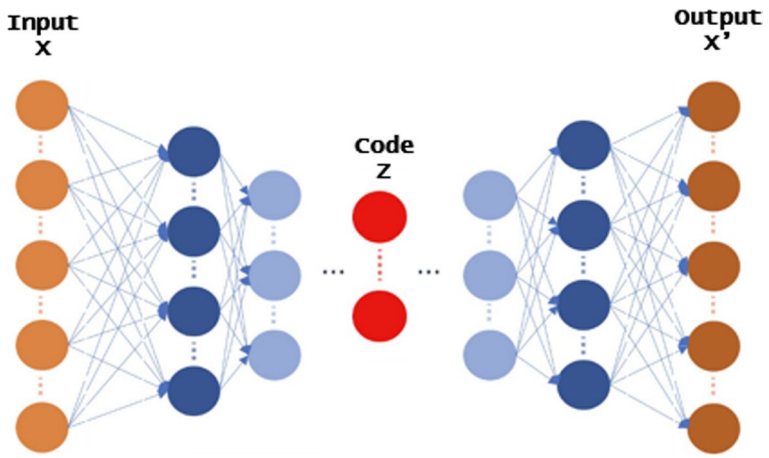

Fig. 2 Schematic structure of an Autoencoder

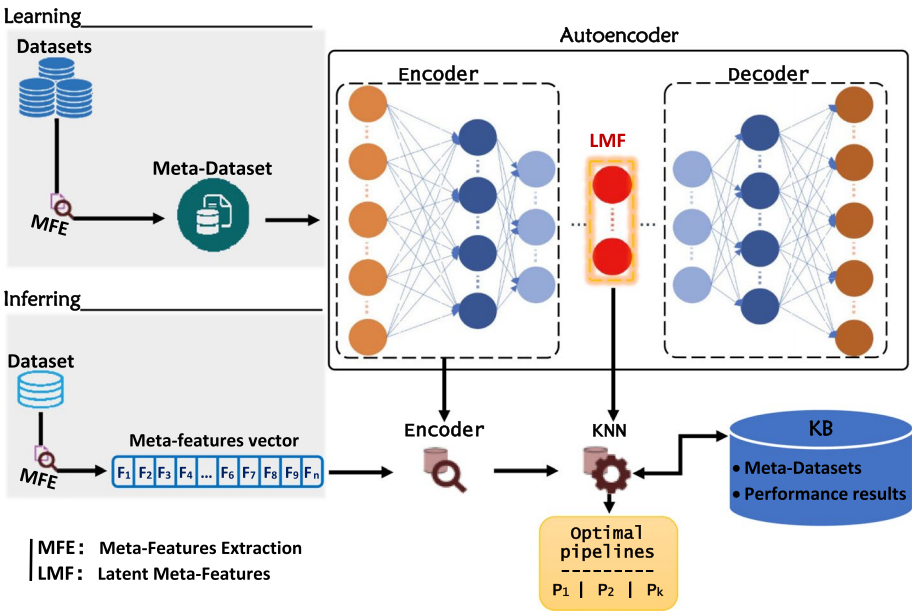

Fig. 3 Overview of proposed AeKNN-based meta-model

Proposed approach

AeKNN consists of two main phases i.e the learning phase and the recommendation phase. The former phase is carried out using the meta-dataset to train the autoencoder model. It allows the extraction of latent meta-features of data. Later, the recommendation phase is performed that principally uses the feed forward autoencoder model which has been generated in the learning phase to extract the latent meta-Features of the test data and, later on, the recommendation and ranking of the optimal pipeline (s) are estimated based on nearest neighbors in the meta-knowledge base. Figure 3 elaborates this process, while Algorithm 1 shows the pseudo-code of AeKNN that is thoroughly discussed in the following.

The proposed methodology constructs an autoencoder which can be used as a latent features extractor. After providing traditional meta-features as input, we train the autoencoder to learn a meaningful latent representation of the meta-dataset. Once the autoencoder is trained, the meta-dataset is forward propagated to extract the latent variables from the middle hidden layer to induce the AeKNN meta-model.

Algorithm 1 AeKNN algorithm's pseudo-code.

Input : Train Data, Test Data, KB /> knowledge base construction
Output : P<P1, P2, P3,. . . , Pn> /> Suggested pipelines
Learning phase :
1: *MetaData* ← MetaFeaturesExtractor(`TrainData`)
2: *AE* ← Autoencoder(`MetaData`)
3: *EncoderModel* ← FeedForwardAEModel(`AE`)
4: *LatentMetaFeatures* ← EncoderModel(`TrainData`)
5: *AeKNN* ← KNN(`LatentMetaFeatures`, KB)
Testing phase :
6: *MetaFeatures* ← MetaFeaturesExtractor(`TestData`)
7: *LatentMetaFeatures* ← EncoderModel(`MetaFeatures`)
8: *OptimalPiplines* ← AeKNN(`LatentMetaFeatures, KB`)

The algorithm consists of two phases. The first phase corresponds to the learning of AeKNN (lines 1–5) while the second phase (lines 6–8) refers to the recommendation phase. During learning, AeKNN focuses on learning a new representation of the data to extract latent meta-features. This is done through the feed forward autoencoder model, using the training meta-data to learn the weights linking the units of autoencoder. During the recommendation phase, the optimal pipelines are generated. The process, performed internally in this phase, transform the extracted meta-features using the autoencoder model which is generated in the training phase (encoder model). It produces a new dataset characterization (latent meta-features), which is more compact representative (line 7) of data. In fact, this new set of features is used by the AeKNN meta-model to recommend the optimal pipeline (s) for the given problem (test dataset) (line 8).

Experimental study

This section describes the experimental design to induce latent meta-features and the evaluation of the proposed approach. In this respect we describe the used datasets, classification algorithms, and meta-Knowledge base construction. Subsequently, the experimental results are presented and discussed in substantial detail.

Datasets, performance evaluation and meta-knowledge base construction

Phase 1 of our experiments includes datasets selection, meta-features extraction, and performance evaluation. We actually collect 400 real-world classification datasets from the popular UCI [1], OpenML [2], Kaggle [3], KEEL [4] repositories and from other real world scenarios. These datasets cover varied tasks with respect to their

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

Table 1 Dataset's dimensions

	Nb. of classes	Nb. of attributes	Nb. of instances
Min	2	5	1368
Max	18	1000	756,400

size, number of attributes, their composition and class imbalance. Datasets characteristiques are illustrated in Table 1.

Later on, in order to build the meta-knowledge base, we evaluate 08 common learning algorithms 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, Stochastic Gradient Descent (SGD) Classifier on each dataset, and recording their generalization performance in terms of Precision, Recall, Accuracy and F1score.

We generate 1000 different combinations of the hyperparameters configurations for each execution of a classifier $\mathcal{C}$ over a dataset $\mathcal{D}$. This process results in an average of 8000 pipelines per dataset. In particular, for each classifier, we generate a list of all possible and reasonable combinations where we conduct, for each dataset, a random search among them. During the training phase, we use a fivefold stratified cross-validation strategy to construct the meta-datasets. As a result, the knowledge base consisted of more than 4 millions evaluated classification pipelines.

In parallel, we extract meta-features of training datasets using the PyMFE tool [52] for the general, statistical, info-theoretical, model-based and landmarking categories. Consequently, it generates a meta-dataset of 400 meta-instances and 60 meta-feature (characteristics) that is used to train the deep autoencoder to extract the latent meta-features. The process of meta-features extraction is formalized by [52] as a function $\mathcal{F} : \mathcal{D} \rightarrow \mathbb{R}^k$ that receives a dataset $\mathcal{D}$ as input, and returns a features vector of k values characterizing the dataset, and that are predictive of algorithms performance when applied to the dataset. Formally, it can be detailed as follows :

$$\mathcal{F}(\mathcal{D}) = \sigma(m(\mathcal{D})) \tag{3}$$

where $\mathcal{D} = \{(x_i, y_i)|i \in \{1, \ldots, N\}\}$ is a dataset with N instances; x_i and y_i indicate the i-th training data and label respectively. The measure $m : \mathcal{D} \rightarrow \mathbb{R}^{k'}$ can extract more than one value from each data set, i.e., k' can vary according to $\mathcal{D}$, which can be mapped to a vector of fixed length k using a summarization function σ. In MtL, where a fixed cardinality is needed, the summarization functions can be, e.g., *mean*, *minimum*, *maximum*, *skewness* and *kurtosis*. Thus, a meta-feature can therefore be seen as a combination of a measure and a summary function [52].

In the recommendation phase, the combination of the meta-dataset and the results of all runs are stored in a meta-knowledge base *KB* where each record represents an execution of a classifier C with hyperparameters configurations H over a dataset D. In particular, each record stores the meta-features that model the dataset, the pipeline, and their interdependencies.

Table 2 Experimental configurations of AeKNN

Model	Number of hidden layers	Number of neurons per layer					Architecture l_i^n
		Layer 1	Layer 2	Latent layer	Layer 4	Layer 5	
AeKNN1	1	–	–	**32**	–	–	(**32**)
AeKNN2	1	–	–	**16**	–	–	(**16**)
AeKNN3	1	–	–	**8**	–	–	(**8**)
AeKNN4	3	32	–	**16**	–	32	(32,**16**,32)
AeKNN5	5	32	16	**8**	16	32	(32,16,**8**,16,32)

The best ones are highlighted in bold

Table 3 List (sample) of benchmark datasets used in the evaluation

Dataset	Number of		
	Instances	Attributes	Classes
APSFailure	76,000	171	2
CustSat	76,020	14	2
car	1728	7	4
kr-vs-kp	3196	37	2
airlines	539383	8	2
vehicle	846	19	4
MiniBooNE	130,064	51	2
jannis	83,733	55	4
nomao	34,465	119	2

Experimental results

AeKNN architectures analysis

AeKNN is characterized by the aforementioned l_i^n parameter that establishes the architecture of the model. This parameter allows the selection of different architectures in term of depth (number of hidden layers) and number of neurons per layer. Table 2 shows the considered architectures. For each model (architecture) the number of hidden layers, as well as the number of neurons in each layer, are shown.

The results produced by the considered architectures using a benchmark of 20 real world datasets with different characteristics (as shown in Table 3) are presented as grouped by classification metrics and datasets. Tables 4, 5 and 6 summarize the evaluation results of each recommended pipeline for each architecture respectively.

The evaluation results of the recommended pipelines by AeKNN with different architectures are presented in Table 4. These results mainly consider the *Accuracy* classification metric. The obtained results indicate that the architectures with single hidden layer perform better on 17 out of 20 datasets, whereas the architectures with three hidden layers perform better on 2 out of 20 datasets while the five hidden layers architecture obtained best results on 1 out of 20 datasets. Therefore, it can be observed through the obtained rankings that single hidden layer architectures perform better most of the times (17) in given circumstances. The $l_i^n = (32)$ works best for most cases (14 win), while $l_i^n = (16)$ has shown disparate results as these can be simultaneously the best values for some cases and worse values for other cases.

Table 4 Accuracy classification results of the recommended pipelines for the considered AeKNN architectures

Dataset	AeKNN				
	(32)	(16)	(8)	(32,16,32)	(32,16,8,16,32)
APSFailure	**0.9921**	0.9734	0.86475	0.9033	0.8325
Higgs	**0.7283**	0.6911	0.4872	0.6398	0.5316
CustSat	0.8155	0.7826	0.5318	**0.8559**	0.6943
car	**0.9999**	0.9808	0.7049	0.9203	0.8277
kr-vs-kp	**0.9976**	0.8130	0.6532	0.7330	0.7291
airlines	0.6982	0.6833	0.5627	**0.7167**	0.4334
vehicle	0.8880	**0.8934**	0.3591	0.8004	0.4098
MiniBooNE	**0.9645**	0.9217	0.8143	0.85	0.7436
jannis	**0.7229**	0.6843	0.6371	0.6911	0.6608
nomao	0.9708	**0.9719**	0.5395	0.6994	0.4659
Credi-g	**0.7921**	0.6502	0.5121	0.3871	0.4768
Kc1	**0.8793**	0.8754	0.3597	0.7488	0.5691
Cnae-9	**0.9671**	0.8923	0.5622	0.5208	0.6049
albert	0.8759	0.8131	0.6981	0.8439	**0.9053**
Numerai28.6	**0.5207**	0.4530	0.3029	0.4760	0.2810
segment	**0.9735**	0.9622	0.8837	0.9508	0.5791
Covertype	**0.8344**	0.7189	0.6521	0.6305	0.4620
KDDCup	**0.9740**	0.8514	0.8034	0.8821	0.8572
shuttle	0.9362	**0.9997**	0.6429	0.8576	0.6744
Gas_Sens-uci	**0.9843**	0.9755	0.7256	0.9667	0.7032

The best performances among all architectures are highlighted in bold

Table 5 F1-Score classification results of the recommended pipelines for the considered AeKNN architectures

Dataset	AeKNN				
	(32)	(16)	(8)	(32,16,32)	(32,16,8,16,32)
APSFailure	0.9823	0.7553	**0.9875**	0.7573	0.9055
Higgs	**0.8743**	0.5451	0.5602	0.4938	0.5316
CustSat	**0.9250**	0.6366	0.4953	0.8194	0.5483
car	0.9635	**0.9874**	0.8144	0.7613	0.6817
kr-vs-kp	**0.9246**	0.7035	0.6532	0.5870	0.8751
airlines	0.5887	**0.7928**	0.5992	0.5707	0.3604
vehicle	0.8515	0.8204	0.2131	**0.9099**	0.3733
MiniBooNE	0.9715	**0.9871**	0.8873	0.7405	0.8531
jannis	0.7229	0.5748	**0.8068**	0.6911	0.6006
nomao	**0.9343**	0.9213	0.5395	0.8454	0.4294
Credi-g	**0.9381**	0.5772	0.5661	0.4141	0.5863
Kc1	0.9321	0.8389	**0.9523**	0.8583	0.4596
Cnae-9	**0.8962**	0.8741	0.6352	0.5938	0.7509
albert	0.8394	0.7036	0.6251	0.8074	**0.9783**
Numerai28.6	0.3747	**0.5260**	0.3029	0.4395	0.3540
segment	**0.9130**	0.8830	0.8837	0.7139	0.5426
Covertype	0.6886	0.6824	**0.7249**	0.4845	0.4620
KDDCup	0.9571	**0.9974**	0.7669	0.8386	0.7112
shuttle	**0.9653**	0.8537	0.4969	0.8306	0.7109
Gas_Sens-uci	0.6161	0.8660	**0.9667**	0.7667	0.8492

The best ones are highlighted in bold

Table 6 AUC classification results of the recommended pipelines for the considered AeKNN architectures

Dataset	AeKNN				
	(32)	(16)	(8)	(32,16,32)	(32,16,8,16,32)
APSFailure	0.9191	**0.9763**	0.8648	0.8639	0.7230
Higgs	0.7283	**0.8371**	0.3412	0.5668	0.5316
CustSat	**0.9654**	0.6731	0.6413	0.8155	0.7673
car	**0.9608**	0.9269	**0.9873**	0.5298	0.6817
kr-vs-kp	0.7765	**0.9103**	0.6167	0.8790	0.5831
airlines	**0.8627**	0.5373	0.6357	0.8442	0.5794
vehicle	**0.9610**	0.8569	0.3956	0.5464	0.5558
MiniBooNE	0.8550	**0.9947**	0.7873	0.7230	0.5976
jannis	**0.7338**	0.7229	0.4911	0.6911	0.5383
nomao	0.8594	0.8423	**0.8978**	0.5899	0.6119
Credi-g	**0.9381**	0.7232	0.5121	0.4601	0.3308
Kc1	0.7333	**0.9119**	0.3962	0.6028	0.6421
Cnae-9	**0.8941**	0.8433	0.4162	0.5938	0.4954
albert	0.9124	**0.9226**	0.6616	0.7344	0.7593
Numerai28.6	**0.6302**	0.5435	0.2664	0.3665	0.2080
segment	0.8900	0.8527	0.6548	0.4362	0.4331
Covertype	0.7979	0.6459	**0.7981**	0.6670	0.4620
KDDCup	**0.9876**	0.7419	0.9408	0.6587	0.7477
shuttle	**0.9727**	0.9267	0.7159	0.9306	0.7839
Gas_Sens-uci	**0.8748**	0.8295	0.7986	0.5572	0.7762

The best ones are highlighted in bold

Table 5 presents the results obtained by AEkNN with the different l_i^n architectures for the *F1-Score* classification metric. Based on the shown findings, there is no ideal architecture for all datasets. In this case, the architecture $l_i^n = (32)$ performs better most of the times (8). Although, the single hidden layer architectures $l_i^n = (16)$ and $l_i^n = (8)$ are the top performers in each of the five cases. Despite being better the same number of times, the architecture $l_i^n = (16)$ has more balanced results.

While, analyzing the classification results corresponding to the *AUC* metric, presented in Table 6, it can be observed that the single hidden layer architectures obtained the best overall results in 20 out of 20 datasets. These rankings show that the single hidden layer model $l_i^n = (32)$ outperformed more often (11), Whereas the $l_i^n = (32, 16, 32)$ and $l_i^n = (32, 16, 8, 16, 32)$ did not deliver any better results.

Therefore, it is considered that $l_i^n = (32)$ is the best among them. Thus, in the following the results of AeKNN, using the presented architecture, is compared against the classical kNN as well as other state-of-the-art meta-models.

To validate and assess the competitiveness provided by the deep autoencoder-KNN based meta-model, we perform a comparative study to other state-of-the-art meta-models with an oversampling approach using the 20-benchmark datasets. We

Table 7 Comparing each baseline meta-model against AeKNN on the 20-benchmark datasets. Listed are the number of datasets where each meta-model produced better predictions than AeKNN *(Wins)*, worse predictions *(Losses)*, or more accurate predictions than all of the other 3 meta-models *(Champion)*

Meta-model	Wins			Losses			Champion		
	Acc	F1-score	AUC	Acc	F1-score	AUC	Acc	F1-score	AUC
AeKNN	-	-	-	-	-	-	16	17	14
KNN	1	2		19	18	17	1	2	3
RF	0	0		20	20	20	0	0	0
XGB	3	1		17	19	17	3	1	3

The best ones are highlighted in bold

compared AeKNN to three widely used meta-models including random forest (RF), k-nearest neighbor (KNN), and XGBoost (XGB) [9, 13, 34, 53].

AeKNN vs traditional meta-models

This section is focused to assess the competitiveness of the proposed meta-model. In this regard, a comparison has been made between the results obtained with AeKNN, using the ${l_i}^n = 32$ architecture as selected in the previous section, and the results obtained with the baseline meta-models on the same datasets. Tables 7 and 8 provide a summarized pairwise one-to-one comparison of the baseline meta-models against AeKNN indicating how often one meta-model is better than another for the three considered classification metrics.

The results shown in Table 7 indicate that the proposed AeKNN works better than most of the traditional meta-models especially the classical kNN and RF for recommending well performing ML pipelines of a given classification dataset. The AeKNN improves kNN in 19 out of 20 cases, obtaining the best overall results in 16 of them and obtains better results than the Random Forest meta-model in all cases for the Accuracy classification metric. Regarding the classification results related to the F1-score and AUC metrics, it can be seen that the ${l_i}^n = 32$ meta-model produces the best overall results in 17 and 14 out of 20 cases respectively, representing an improvement of 17 times over the KNN.

As shown in Table 8 and summarized in 7, the results obtained through the proposed AeKNN meta-model improve those obtained with the traditional kNN as well as the other state of the art meta-models for most of the datasets for the automated selection of ML algorithms.

Conclusion

In this paper, a novel meta-model based latent features extraction method, namely AeKNN, is proposed. This model is based on kNN to recommend the optimal pipelines while its major objective is to mitigate the data characterization limitations. In this regard, AeKNN internally incorporates a model-building phase which is aimed at an extraction of latent meta-features, using a feed forward autoencoder. The main reason that has led to the design of AeKNN is the good results that have been obtained

Table 8 Results of the RF, XGB, KNN, AeKNN meta-models for recommending optimal pipelines for test data

Dataset	AeKNN			KNN			XGB			RF		
	Acc	F1-score	AUC	Acc	F1-score	AUC	Acc	F1-score	AUC	Acc	F1-score	AUC
APSFailure	**0.9921**	**0.9823**	0.9191	0.991	0.9105	0.7150	0.9673	0.8868	**0.9468**	0.8950	0.6920	0.6920
Higgs	**0.7283**	**0.8743**	**0.7283**	0.7260	0.5725	0.4865	0.6801	0.5631	0.4771	0.6072	0.5867	0.5867
CustSat	0.8155	**0.9250**	**0.9654**	0.7998	0.7558	0.7063	**0.8715**	0.5355	0.6320	0.7382	0.7542	0.7542
car	**0.9999**	**0.9635**	0.7724	0.9754	0.7124	**0.9608**	0.9462	0.6467	0.8162	0.8549	0.6884	0.6884
kr-vs-kp	**0.9976**	**0.9246**	0.7631	0.9209	0.7309	0.6449	0.7593	0.4963	**0.7765**	0.6532	0.4867	0.4867
airlines	0.6982	0.5887	**0.8627**	0.6758	0.5953	0.5823	**0.7094**	**0.6289**	0.5429	0.5927	0.3532	0.3532
vehicle	0.888	**0.8515**	**0.9610**	0.8415	0.7610	0.7480	**0.9027**	0.6762	0.8822	0.6591	0.6386	0.6386
MiniBooNE	**0.9645**	**0.9715**	**0.8550**	0.9423	0.7888	0.7393	0.8903	0.8098	0.6143	0.8343	0.5583	0.5583
jannis	**0.7229**	**0.7229**	**0.7338**	0.6719	0.5549	0.6879	0.6845	0.4215	0.4815	0.6171	0.4506	0.4506
nomao	**0.9708**	**0.9343**	0.8007	0.9570	0.6940	**0.8594**	0.7987	0.6817	0.5227	0.6995	0.4965	0.4965
Credi-g	**0.7921**	**0.9381**	**0.9381**	0.7188	0.4923	0.6983	0.5739	0.5299	0.5899	0.6121	0.5916	0.5916
Kc1	**0.8793**	**0.9321**	**0.7333**	0.8552	0.6287	0.6887	0.7697	0.4337	0.7127	0.7097	0.6892	0.6892
Cnae-9	**0.9671**	0.7998	**0.8941**	0.8803	**0.8962**	0.7868	0.8365	0.6830	0.8525	0.7922	0.5892	0.5892
albert	**0.8759**	**0.8394**	**0.9124**	0.8005	0.7565	0.6340	0.8288	0.6753	0.5528	0.7981	0.7046	0.7046
Numerai28.6	**0.5207**	**0.3747**	**0.6302**	0.4433	0.1803	0.4228	0.4836	0.2936	0.2076	0.4229	0.2971	0.3571
segment	**0.9735**	**0.9130**	**0.8900**	0.9681	0.7416	0.7651	0.9542	0.7642	0.8242	0.9337	0.8767	0.8767
Covertype	**0.8344**	**0.6886**	0.7204	0.7307	0.6502	0.6007	0.7890	0.4530	**0.7592**	0.6521	0.4126	0.4126
KDDCup	**0.9740**	**0.9571**	0.9660	0.9500	0.8330	**0.9876**	0.9331	0.7796	0.9491	0.8934	0.7634	0.7634
shuttle	0.9362	**0.9653**	**0.9727**	**0.9905**	0.9100	0.9700	0.9649	0.6289	0.6889	0.8429	0.6764	0.6764
Gas_Sens-uci	**0.9843**	0.6161	**0.8748**	0.9739	**0.8569**	0.6979	0.9468	0.8298	0.6708	0.9256	0.7591	0.7591

The results for all meta-models are presented jointly, and the best ones are highlighted in bold

by autoencoder when they are used to generate higher-level features and those of KNN for performing pipelines recommendation in meta-learning systems. AeKNN relies on a feed forward autoencoder to extract latent representations of a higher level that replaces the original meta-features.

In order to assess the competitiveness of the proposed approach, a series of experiments are carried out. Initially, the analysis of results have allowed to determine the architecture of autoencoder. Furthermore, in the later parts of the conducted experiments, the results of the adopted architecture have been compared with the results produced by the state-of-the-art meta-models. It is observed that AeKNN offers a considerable improvement of the results obtained by all baseline meta-models. These results show that the use of autoencoders can be helpful to extract relevant meta-features which are more significant and informative. It thus improve the effectiveness of meta-learning, and broadens the directions of future work. They can be applied to support the solution of similar problems, in a better manner than the traditional meta-models.

Abbreviations

ML	Machine learning
AeKNN	Autoencoder-kNN
Ae	Autoencoder
kNN	k-Nearest Neighbors
AutoML	Automated Machine Learning
ASP	Algorithm Selection Problem
PCA	Principal Component Analysis
MLP	Multi-Layer Perceptron
SVC	Support Vector Classifier
SGD	Stochastic Gradient Descent
RF	Random Forest

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'Université du Littoral Côte d'Opale) for the computational facilities.

Author contributions
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.

Received: 9 December 2021 Accepted: 21 January 2023
Published online: 03 February 2023

References

1. Garouani M, Ahmad A, Bouneffa M, Hamlich M, Bourguin G, Lewandowski A. Using meta-learning for automated algorithms selection and configuration: an experimental framework for industrial big data. J Big Data. 2022. https://doi.org/10.1186/s40537-022-00612-4.

2. Adadi A. A survey on data-efficient algorithms in big data era. J Big Data. 2021;8(1):24. https://doi.org/10.1186/s40537-021-00419-9.
3. Rostami M, Berahmand K, Forouzandeh S. A novel community detection based genetic algorithm for feature selection. J Big Data. 2020;8(1):2. https://doi.org/10.1186/s40537-020-00398-3.
4. 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.
5. Feurer M, Klein A, Eggensperger K, Springenberg JT, Blum M, Hutter F. Efficient and robust automated machine learning. In: Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2. NIPS'15, pp. 2755–2763. MIT Press.
6. Garouani M, Ahmad A, Bouneffa M, Hamlich M, Bourguin G, Lewandowski A. Towards big industrial data mining through explainable automated machine learning. Int J Advan Manuf Technol. 2022;120(1–2):1169–88. https://doi.org/10.1007/s00170-022-08761-9.
7. Garouani M, Ahmad A, Bouneffa M, Hamlich M. Scalable Meta-Bayesian Based Hyperparameters Optimization for Machine Learning. In: Hamlich M, Bellatreche L, Siadat A, Ventura S, editor. Smart Applications and Data Analysis. SADASC 2022. Communications in Computer and Information Science, vol 1677. Cham: Springer; 2022. https://doi.org/10.1007/978-3-031-20490-6_14
8. 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: 23rd International Conference on Enterprise Information Systems, pp. 709–716. https://doi.org/10.5220/0010457107090716.
9. Laadan D, Vainshtein R, Curiel Y, Katz G, Rokach L. RankML: a Meta Learning-Based Approach for Pre-Ranking Machine Learning Pipelines. 2019. 1911.00108.
10. Garouani M, Zaysa K. Leveraging the Automated Machine Learning for Arabic Opinion Mining: A Preliminary Study on AutoML Tools and Comparison to Human Performance. In: Motahhir S, Bossoufi B, editor. Digital Technologies and Applications. ICDTA 2022. Lecture Notes in Networks and Systems, vol 455. Cham: Springer; 2022. https://doi.org/10.1007/978-3-031-02447-4_17
11. Thornton C, Hutter F, Hoos HH, Leyton-Brown K. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '13, pp. 847–855. Association for Computing Machinery. https://doi.org/10.1145/2487575.2487629.
12. Olson RS, Moore JH. TPOT: A Tree-Based Pipeline Optimization Tool for Automating Machine Learning. In: Hutter F, Kotthoff L, Vanschoren J (eds.) Automated Machine Learning: Methods, Systems, Challenges. The Springer Series on Challenges in Machine Learning, pp. 151–160. Springer International Publishing. https://doi.org/10.1007/978-3-030-05318-5_8.
13. Garouani M, Hamlich M, Ahmad A, Bouneffa M, Bourguin G, Lewandowski A. Toward an automatic assistance framework for the selection and configuration of machine learning based data analytics solutions in industry 4.0. In: Proceedings of the 5th International Conference on Big Data and Internet of Things, pp. 3–15. Springer. https://doi.org/10.1007/978-3-031-07969-6_1.
14. Nural MV, Peng H, Miller JA. 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.
15. Garouani M, Ahmad A, Bouneffa M, Hamlich M, Bourguin G, Lewandowski A. Towards meta-learning based data analytics to better assist the domain experts in industry 4.0. In: Artificial Intelligence in Data and Big Data Processing, pp. 265–277. Springer. https://doi.org/10.1007/978-3-030-97610-1_22.
16. Peng Y, Flach PA, Soares C, Brazdil P. Improved Dataset Characterisation for Meta-learning. In: Lange S, Satoh K, Smith CH, editors. Discovery Science. Lecture Notes in Computer Science. Springer: Berlin; 2002. p. 141–52. https://doi.org/10.1007/3-540-36182-0_14.
17. Vanschoren J. Meta-Learning: A Survey. arxiv:1810.03548.
18. Matejka J, Fitzmaurice G. Same Stats, Different Graphs: Generating Datasets with Varied Appearance and Identical Statistics through Simulated Annealing. In: Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, pp. 1290–1294. Association for Computing Machinery. https://doi.org/10.1145/3025453.3025912.
19. Kalousis A, Hilario M. Feature Selection for Meta-learning. In: Cheung D, Williams GJ, Li Q, editors. Advances in Knowledge Discovery and Data Mining. Lecture Notes in Computer Science. Berlin: Springer; 2001. p. 222–33. https://doi.org/10.1007/3-540-45357-1_26.
20. Pavel Y, Soares BC. Decision tree-based data characterization for meta-learning. IDDM. 2002;111:78.
21. Meskhi MM, Rivolli A, Mantovani RG, Vilalta R. Learning Abstract Task Representations. arxiv:2101.07852.
22. Alzubaidi L, Zhang J, Humaidi AJ, Al-Dujaili A, Duan Y, Al-Shamma O, Santamaría J, Fadhel MA, Al-Amidie M, Farhan L. Review of deep learning: Concepts, CNN architectures, challenges, applications, future directions. J Big Data. 2021;8(1):53. https://doi.org/10.1186/s40537-021-00444-8.
23. Deng L, Yu D. Deep learning: methods and applications. Found Trends Signal Processing. 2014;7(34):197–387. https://doi.org/10.1561/2000000039.
24. Gosztolya G, Busa-Fekete R, Grósz T, Tóth L. DNN-Based Feature Extraction and Classifier Combination for Child-Directed Speech, Cold and Snoring Identification. In: Interspeech 2017, pp. 3522–3526. ISCA. https://doi.org/10.21437/Interspeech.2017-905.
25. Wang W, Huang Y, Wang Y, Wang L. Generalized Autoencoder: A Neural Network Framework for Dimensionality Reduction. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 496–503. https://doi.org/10.1109/CVPRW.2014.79.
26. Bhatia V, Rani R. DFuzzy: A deep learning-based fuzzy clustering model for large graphs. Knowl Inform Syst. 2018;57(1):159–81. https://doi.org/10.1007/s10115-018-1156-3.
27. Vincent P, Larochelle H, Bengio Y, Manzagol P-A. Extracting and composing robust features with denoising autoencoders. In: Proceedings of the 25th International Conference on Machine Learning. ICML '08, pp. 1096–1103. Association for Computing Machinery. https://doi.org/10.1145/1390156.1390294.

28. Pulgar FJ, Charte F, Rivera AJ, del Jesus MJ. AEkNN: An AutoEncoder kNN-Based Classifier With Built-in Dimensionality Reduction. Int J Comput Intell Syst. 2018;12(1):436–52. https://doi.org/10.2991/ijcis.2018.125905686.
29. Muñoz MA, Sun Y, Kirley M, Halgamuge SK. Algorithm selection for black-box continuous optimization problems: A survey on methods and challenges. Information Sci. 2015;317:224–45. https://doi.org/10.1016/j.ins.2015.05.010.
30. Feurer M, Springenberg J, Hutter F. Initializing Bayesian Hyperparameter Optimization via Meta-Learning. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 29. 2015. https://ojs.aaai.org/index.php/AAAI/article/view/9354.
31. Drori I, Krishnamurthy Y, Lourenço R, Rampin R, Cho K, Silva C, Freire J. Automatic Machine Learning by Pipeline Synthesis Using Model-Based Reinforcement Learning and a Grammar. arxiv:1905.10345
32. Li L, Jamieson KG, DeSalvo G, Rostamizadeh A, Talwalkar A. Efficient hyperparameter optimization and infinitely many armed bandits. CoRR abs/1603.06560 (2016). arxiv:1603.06560.
33. das Dôres SN, Alves L, Ruiz DD, Barros RC. A meta-learning framework for algorithm recommendation in software fault prediction. In: Proceedings of the 31st Annual ACM Symposium on Applied Computing. SAC '16, pp. 1486–1491. Association for Computing Machinery. https://doi.org/10.1145/2851613.2851788.
34. Cohen-Shapira N, Rokach L, Shapira B, Katz G, Vainshtein R. AutoGRD: Model Recommendation Through Graphical Dataset Representation. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management. CIKM '19, pp. 821–830. https://doi.org/10.1145/3357384.3357896.
35. Reif M, Shafait F, Goldstein M, Breuel T, Dengel A. Automatic classifier selection for non-experts. Pattern Anal Appl. 2012. https://doi.org/10.1007/s10044-012-0280-z.
36. Pinto F, Soares C, Mendes-Moreira Ja. Towards Automatic Generation of Metafeatures. In: PAKDD. https://doi.org/10.1007/978-3-319-31753-3_18.
37. Katz G, Shin EC, Song D. ExploreKit: Automatic Feature Generation and Selection. 2016 IEEE 16th International Conference on Data Mining (ICDM). https://doi.org/10.1109/ICDM.2016.0123.
38. Vainshtein R, Greenstein-Messica A, Katz G, Shapira B, Rokach L. A Hybrid Approach for Automatic Model Recommendation. In: Proceedings of the 27th ACM International Conference on Information and Knowledge Management. CIKM '18, pp. 1623–1626. https://doi.org/10.1145/3269206.3269299.
39. Vilalta R, Drissi Y. A Perspective View and Survey of Meta-Learning. Artif Intell Rev. 2002;18(2):77–95. https://doi.org/10.1023/A:1019956318069.
40. Sagi O, Rokach L. Ensemble learning: A survey. WIREs Data Mining and Knowledge Discovery. 2018;8(4):1249. https://doi.org/10.1002/widm.1249.
41. Santoro A, Bartunov S, Botvinick M, Wierstra D, Lillicrap T. Meta-learning with memory-augmented neural networks. In: Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48. ICML'16, pp. 1842–1850. JMLR.org.
42. Bruno A. Pimentel and André Carlos Ponce de Leon Ferreira de Carvalho: A new data characterization for selecting clustering algorithms using meta-learning. Inform Sci. 2018;477:203–19. https://doi.org/10.1016/j.ins.2018.10.043.
43. Rendell L, Seshu R, Tcheng D. Layered concept-learning and dynamically-variable bias management. In: Proceedings of IJCAI-87, pp. 308–314. Morgan Kaufmann.
44. Pfahringer B. Tell me who can learn you and I can tell you who you are: Landmarking Various Learning Algorithms. https://www.semanticscholar.org/paper/Tell-me-who-can-learn-you-and-I-can-tell-you-who-Pfahringer-Bensusan/78e71a6a649dd6778bb1c0923f626d6573cc2b06.
45. Michie D, Spiegelhalter DJ, Taylor CC, Campbell J (eds). Machine Learning, Neural and Statistical Classification. Ellis Horwood. 1995.
46. Souza BF. Meta-aprendizagem Aplicada à Classificação de Dados de Expressão Gênica. https://doi.org/10.11606/T.55.2010.tde-04012011-142551.
47. Ferrari DG, de Castro LN. Clustering algorithm selection by meta-learning systems: A new distance-based problem characterization and ranking combination methods. Inform Sci. 2015;301:181–94. https://doi.org/10.1016/j.ins.2014.12.044.
48. Yaveroglu ON, Malod-Dognin N, Davis D, Levnajic Z, Janjic V, Karapandza R, Stojmirovic A, Pržulj N. Revealing the Hidden Language of Complex Networks. Sci Rep. 2014;4(1):4547. https://doi.org/10.1038/srep04547.
49. Bilalli B, Abello A, Aluja-Banet T. On the predictive power of meta-features in OpenML. Int J Appl Math Computer Sci. 2017;27(4):697–712. https://doi.org/10.1515/amcs-2017-0048.
50. Hotelling H. Analysis of a complex of statistical variables into principal components. J Educ Psychol. 1993;24(6):417–41. https://doi.org/10.1037/h0071325.
51. Hancock JT, Khoshgoftaar TM. Survey on categorical data for neural networks. J Big Data. 2020;7(1):28. https://doi.org/10.1186/s40537-020-00305-w.
52. Alcobaça E, Siqueira F, Rivolli A, Garcia LPF, Oliva JT, de Carvalho ACPLF. MFE: Towards reproducible meta-feature extraction. J Mach Learning Res 2020;**21**(111), 1–5.
53. Cohen-Shapira N, Rokach L. Automatic Selection of Clustering Algorithms Using Supervised Graph Embedding. arxiv:2011.08225.

Publisher's Note