Towards Efficient and Explainable Automated Machine Learning Pipelines Design

Moncef GAROUANI

Institut de Recherche en Informatique de Toulouse
26 May 2023

Moncef Garouani
Email : mgarouani@gmail.com
Website : www.mgarouani.fr
Temporary Lecturer and Research Assistant
EILCO /ULCO University - LISIC Laboratory

Slides available at mgarouani.fr/talks → IRIT seminar
(all references are clickable links)

Outline

1 Context

2 Problem Statement and the State of the art

3 Research work
- Towards a Meta-learning based AutoML framework for Industrial big data
- Learning abstract tasks representation
- Towards interactive explainable AutoML
- AMLBID : a self-explainable AutoML software package

4 Conclusion & perspectives

1 Context

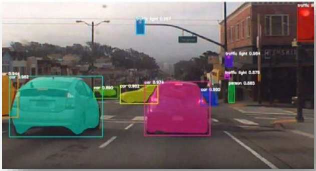
2 Problem Statement and the State of the art

3 Research work
- Towards a Meta-learning based AutoML framework for Industrial big data
- Learning abstract tasks representation
- Towards interactive explainable AutoML
- AMLBID : a self-explainable AutoML software package

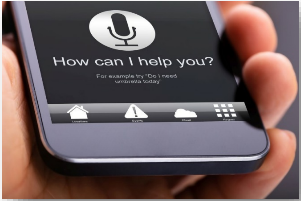
4 Conclusion & perspectives

Successes of Machine & Deep learning

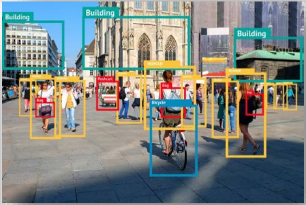
Self-driving cars



Robotic



Speech recognition



Objects recognition

Machine Learning solutions in the industry

Advantages

- ✚ High predictive accuracy
- ✚ Data-driven, few assumptions

Challenges

- ✘ Various ML algorithms : **Which one to choose?**
- ✘ Numerous Hyperparameters (categorical, continuous, conditional)
- ✘ Numerous metrics of performance (Acc, AUC, Recall, etc.)
- ✘ **Need high technical expertise** in statistics and data science

Machine Learning solutions in the industry

Advantages

- ✚ High predictive accuracy
- ✚ Data-driven, few assumptions

Challenges

- ✘ Various ML algorithms : **Which one to choose?**
- ✘ Numerous Hyperparameters (categorical, continuous, conditional)
- ✘ Numerous metrics of performance (Acc, AUC, Recall, etc.)
- ✘ **Need high technical expertise** in statistics and data science

Accuracy	**[Mazumder *et al.*]**	**[Tarak *et al.*]**	**CNC MTW**	**[Thiyagu, *et al.*]**
Best ML algorithm	**0.93**	**0.99**	**0.78**	**0.97**
	Grad. Boosting	**DT**	**SVM**	**RF**
Best Manufacturing Score	0.85	0.98	0.62	0.92

⇝ **No "one-size-fits-all" ML solution for advanced analytics**

Developing advanced Analytics : Goal

Many scenarios, many variables, vorious needs
Continuously changing!

→

Brut-force selection of ML methods and design parameters
Prohibitively expensive & require technical expertise

→

Make Machine Learning
Do the Crafting

Developing advanced Analytics : Goal

Many scenarios, many variables, vorious needs
Continuously changing!

→

Brut-force selection of ML methods and design parameters
Prohibitively expensive & require technical expertise

→

Make Machine Learning
Do the Crafting

Mission statement

Enabling users to efficiently apply ML!
$\rightsquigarrow$ Develop **holistic transparent AutoML**

1 Context

2 Problem Statement and the State of the art

3 Research work
- Towards a Meta-learning based AutoML framework for Industrial big data
- Learning abstract tasks representation
- Towards interactive explainable AutoML
- AMLBID : a self-explainable AutoML software package

4 Conclusion & perspectives

The algorithms selection and configuration problem

Definition : Combined Algorithms selection and Hyperparameters optimization (CASH)

Given :

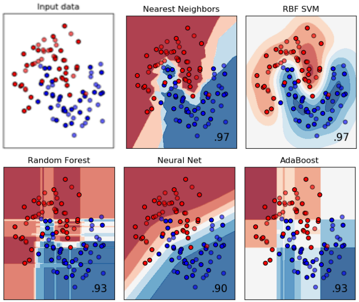
- a set of algorithms $\mathcal{A} = \{A^{(1)}, \ldots, A^{(n)}\}$
- $\mathcal{H}^{(i)}$ the hyperparameters space of $A^{(i)} i \in 1, \ldots, n$
- a set of training problem instances $\mathcal{D}$ divided on D_{train} and D_{valid}
- a cost metric $\mathcal{L} : A^{(i)} \times H_n \times D \to \mathbb{R}$ assessing the predictive performance of the model induced by the algorithm $A^{(i)}$ with an HP configuration $H_n \in \mathcal{H}^{(i)}$ on the dataset D

Find : $A^{(i)}_{H*}$ that minimizes or maximizes the $\mathcal{L}$ on $\mathcal{D}$ such that :

$$A^{(i)}_{H*} \in \underset{A^{(i)} \in \mathcal{A}, H \in \mathcal{H}}{argmin} \mathcal{L}(A_H, D_{train}, D_{validation})$$

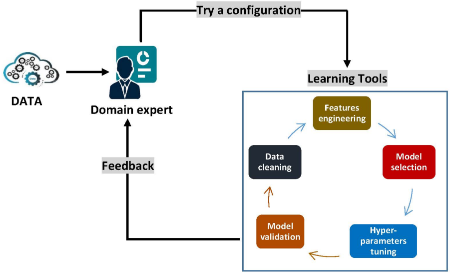
Challenges of the algorithms selection and configuration

1. A pool of ML algorithms to be tested
2. Loop over all candidate pipelines
3. Instantiate and evaluate the ML model based on each pipeline
4. Select the best ML model based on the performance



Challenges of the algorithms selection and configuration

1. A pool of ML algorithms to be tested
2. Loop over all candidate pipelines
3. Instantiate and evaluate the ML model based on each pipeline
4. Select the best ML model based on the performance



The blackbox function is **expensive** to be evaluated
⇝ It is important to automate the Algorithms selection and configuration process

Automated Machine Learning

Definition : Automated Machine Learning (AutoML)

- Automated machine learning is the process of applying ML models to real-world problems using automation.
- It automates the selection, composition and parameterization of ML models.
- AutoML makes ML techniques accessible to domain scientists who are interested in applying advanced analytic but lack the required expertise.
- This can be seen as a democratization of ML.

Automated Machine Learning

Definition : Automated Machine Learning (AutoML)

- Automated machine learning is the process of applying ML models to real-world problems using automation.
- It automates the selection, composition and parameterization of ML models.
- AutoML makes ML techniques accessible to domain scientists who are interested in applying advanced analytic but lack the required expertise.
- This can be seen as a democratization of ML.

Objectives

- Automatic selection of algorithms
- Automatic tuning of hyperparameters
- Solve the CASH

Benefits

- Reduce the required expertise
- Faster development of algorithms
- Less human time
- Further automation

AutoML as a CASH problem

AutoML

Given a training set $\mathcal{D}$ and a set of algorithms $\mathcal{A}$ with an associated hyperparameters space $\mathcal{H}$, the AutoML for the CASH problem is to find the optimal algorithm and hyperparameters space combination $(A^{(i)}, H^{*})$ that minimize or maximize the coast metric $\mathcal{L}$ evaluated on a validation set $\mathcal{D}_{validation}$.

$$A_{H*}^{(i)} \in \underset{A^{(i)} \in \mathcal{A}, H \in \mathcal{H}}{argmin} \mathcal{L}(A_H, D_{train}, D_{validation})$$

AutoML as a CASH problem

AutoML

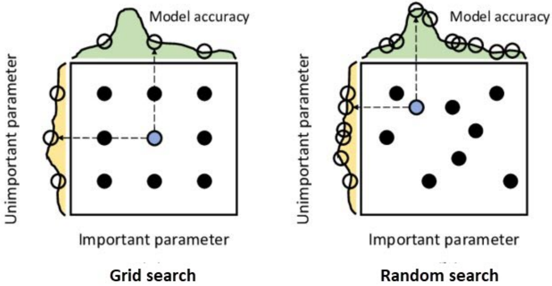
Given a training set $\mathcal{D}$ and a set of algorithms $\mathcal{A}$ with an associated hyperparameters space $\mathcal{H}$, the AutoML for the CASH problem is to find the optimal algorithm and hyperparameters space combination $(A^{(i)}, H^*)$ that minimize or maximize the coast metric $\mathcal{L}$ evaluated on a validation set $\mathcal{D}_{validation}$.

$$A^{(i)}_{H*} \in \underset{A^{(i)} \in \mathcal{A}, H \in \mathcal{H}}{argmin} \mathcal{L}(A_H, D_{train}, D_{validation})$$

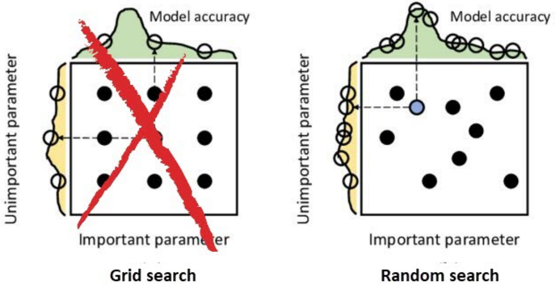
How to search?

- Grid & Random search
- Bayesian optimization [AutoSklearn]
- Evolutionary algorithms [TPOT]
- Meta-learning (Largely unexplored)

Grid Search and Random Search



Grid Search and Random Search

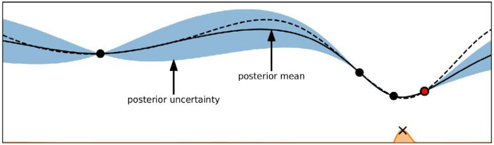


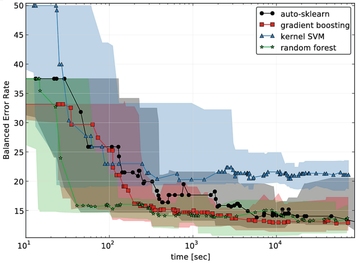
- Both completely uninformed [Bergstra *et al.* (2012)]
- Grid search suffers from the curse of dimensionality [Bergstra *et al.* (2012)]
- Random search handles low intrinsic dimensionality better [Andradóttir *et al.* (2015)]

Bayesian Optimization

Autosklearn [Feurer *et al.* (2019, 2020)]

- Start with few (random or guided) HPs configurations
- Repeat until stopping criterion (fixed budget, convergence, etc.)
- Accurate but so expensive and can overfits easily

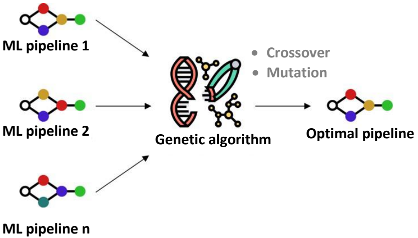


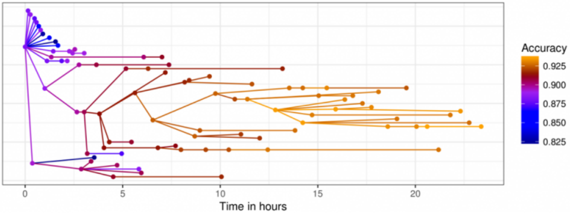


Genetic algorithms

Tree-based Pipeline Optimization Tool (TPOT) [Oslon *et al.* (2016)]

- Start with random pipelines; best of every generation will cross-over or mutate
- Pipelines are represented by a tree of unlimited length and depth
- Accurate but so expensive and could generate invalid individuals





Observations and main ideas

Observations

- Obs 1 : We cannot afford to evaluate all configurations $H \in \mathcal{H}$ on all instances $\mathcal{I} \in \mathcal{D}$
- Obs 2 : We do not want to waste time on less performing H_n values
- Obs 3 : We need enough empirical evidence to distinguish between well performing $(A^{(i)}, H)$
- Obs 4 : Algorithms configuration can lead to over-tuning
- Obs 5 : If done wrong, waste of time and compute resources

Idea

- Idea 1 : Discard less performing (A, H_n) early on
- Idea 2 : Transfer knowledge when optimizing on new tasks
- Idea 3 : Guide the optimization process

Towards human-like learning to learn

Humans learn across tasks

Why? Requires less trial-and-error, less data and time

When one learn new skills, (s)he rarely, if ever, starts from scratch.

- Start from skills learned earlier in related tasks.
- Reuses approaches that worked well before, and focuses on what is likely worth trying based on experience.
- With every learned skill, learning new skills becomes easier, requiring fewer examples and less trial-and-error.

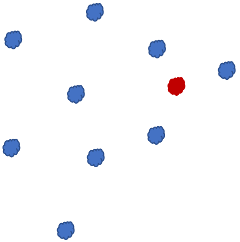
In short, **we learn how to learn across tasks**

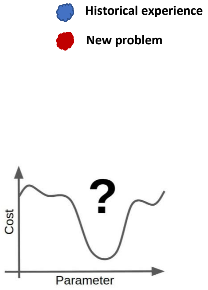
Beyond blackbox optimization

Idea : Based on the assumption "*Algorithms show similar performance with the same configuration for similar problems*" $\rightsquigarrow$ Take the best configurations from previous runs and try them as initial design on new instances.

Beyond blackbox optimization

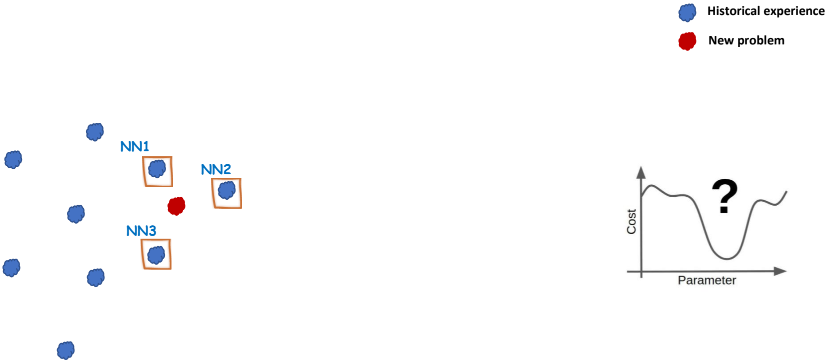
Idea : Based on the assumption "*Algorithms show similar performance with the same configuration for similar problems*" ⇝ Take the best configurations from previous runs and try them as initial design on new instances.





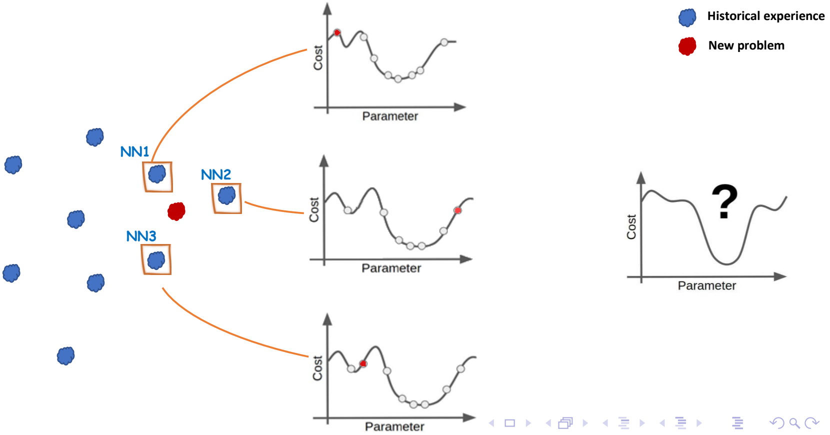
Beyond blackbox optimization

Idea : Based on the assumption "*Algorithms show similar performance with the same configuration for similar problems*" ⇝ Take the best configurations from previous runs and try them as initial design on new instances.



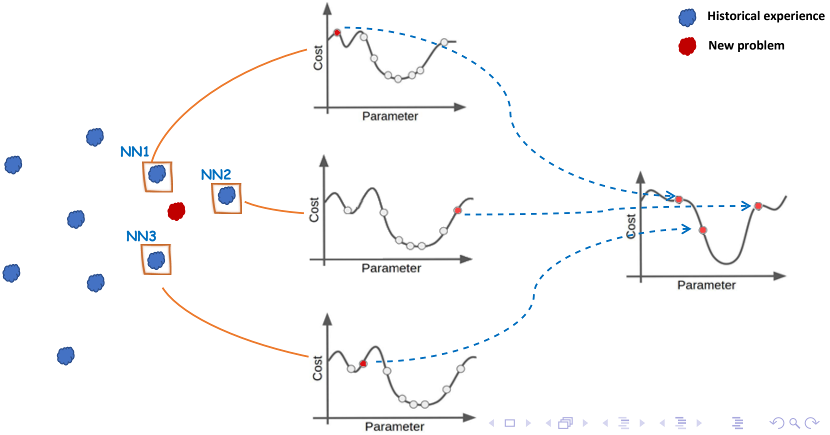
Beyond blackbox optimization

Idea : Based on the assumption "*Algorithms show similar performance with the same configuration for similar problems*" ⇝ Take the best configurations from previous runs and try them as initial design on new instances.



Beyond blackbox optimization

Idea : Based on the assumption "*Algorithms show similar performance with the same configuration for similar problems*" ⇝ Take the best configurations from previous runs and try them as initial design on new instances.



1 Context

2 Problem Statement and the State of the art

3 Research work
- Towards a Meta-learning based AutoML framework for Industrial big data
- Learning abstract tasks representation
- Towards interactive explainable AutoML
- AMLBID : a self-explainable AutoML software package

4 Conclusion & perspectives

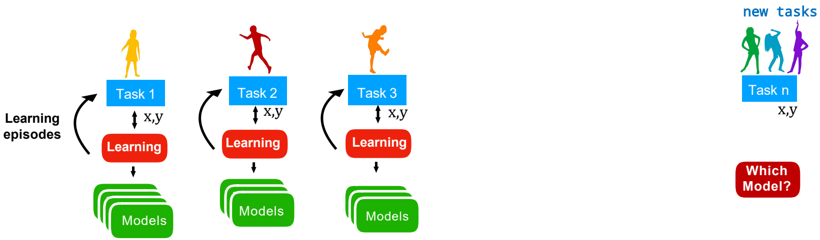
1 Context

2 Problem Statement and the State of the art

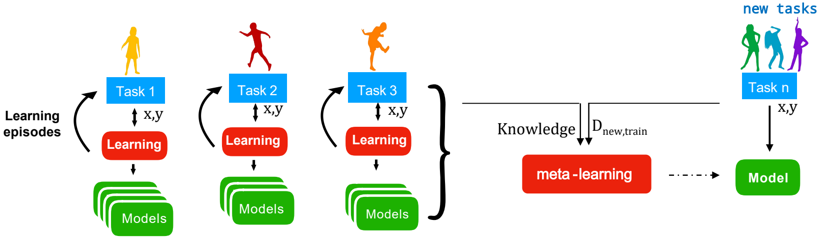
3 Research work
- Towards a Meta-learning based AutoML framework for Industrial big data
- Learning abstract tasks representation
- Towards interactive explainable AutoML
- AMLBID : a self-explainable AutoML software package

4 Conclusion & perspectives

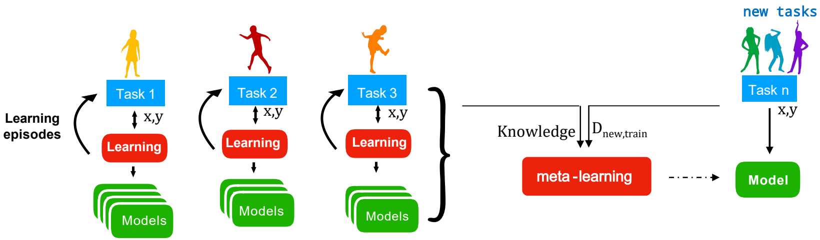
Learning is a never-ending process



Learning is a never-ending process

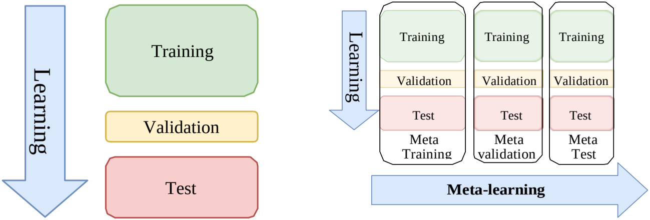


Learning is a never-ending process



Learn more effectively : less trial-and-error, less data, and less time

Meta-learning



(a) Learning **(b) Meta-learning**

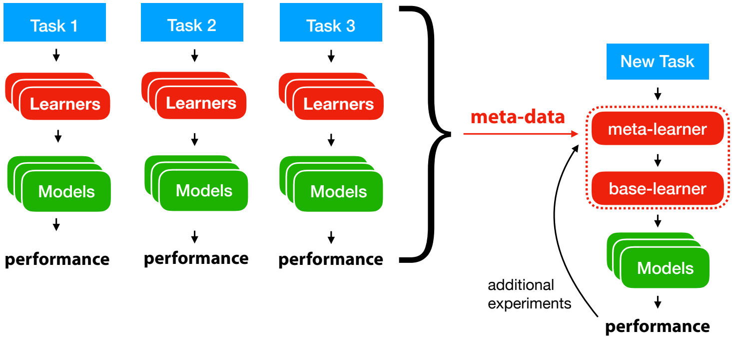
Source : OBOE [Yang *et al.*, 2019]

We can use meta-learning to generalize across datasets and models by :

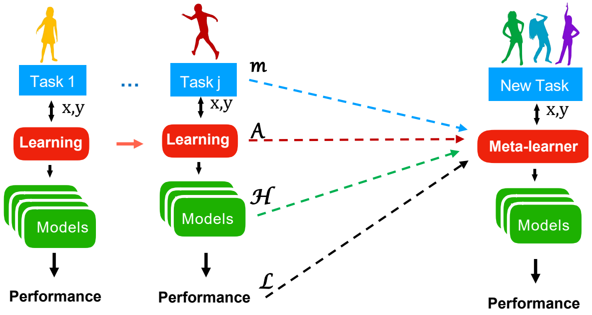
- Learning which hyperparameters are really important
- Learning which hyperparameters values should be tried first
- Learning which architectures will most likely work

Meta-learning in practice

We need a meta-data repository of relevant prior machine learning experiments to transfer prior knowledge across tasks.



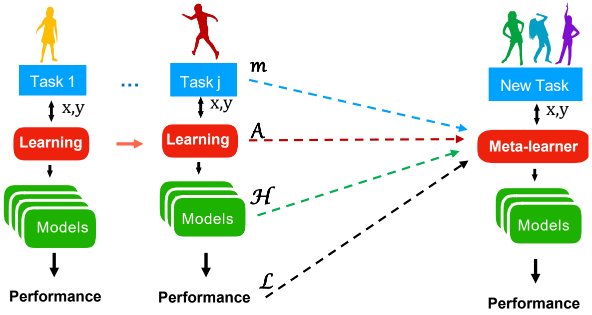
Meta-data



Notation reminder

- m : Meta-features
- $\mathcal{A}$: Learning algorithms
- $\mathcal{H}$: Models Hyperparameters
- $\mathcal{L}$: Performance estimate of $\mathcal{A}(\mathcal{H})$ on the Task j

Meta-data



Notation reminder

- m : Meta-features
- $\mathcal{A}$: Learning algorithms
- $\mathcal{H}$: Models Hyperparameters
- $\mathcal{L}$: Performance estimate of $\mathcal{A}(\mathcal{H})$ on the Task j

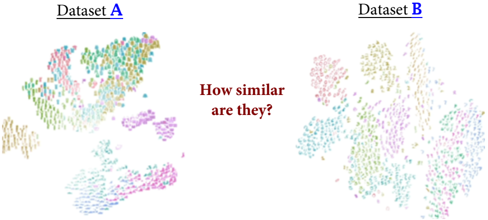
But how can we **featurize** a task (dataset)?

Meta-learning

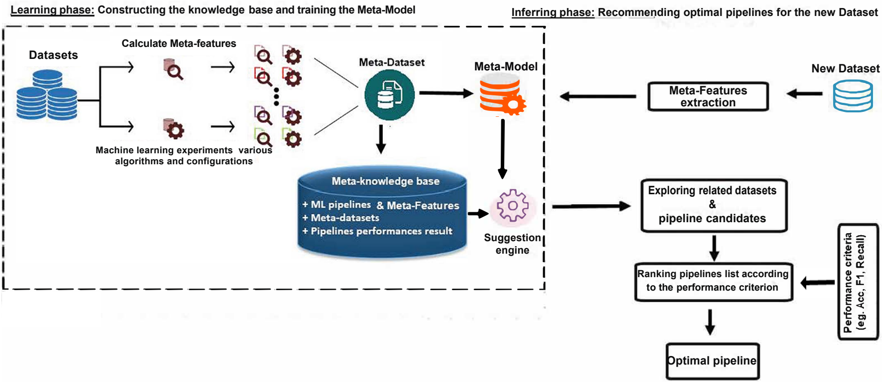
How to measure tasks similarity?

Tasks similarity

- **Statistical** meta-features that describe tabular datasets [Vanschoren *et al.* (2018)]
- **Task2Vec** : task embedding for image data [Achille *et al.* (2019)]
- **Optimal transport** : similarity measure based on comparing probability distributions [Alvarez-Meliset *al.* (2020)]
- **Metadata embedding** based on textual dataset description [Drori *et al.* (2019)]
- **Dataset2Vec** : compares batches of datasets [Jooma *et al.* (2020)]



Conceptual description



Prototypical implementation

AMLBID

- 400 CASH scenarios from I4.0 AI domains

Datasets

- 400 real-world classification datasets
- Mix of binary (71%) and multiclass (29%)
- Process, Machine & Supply chain tasks

	Classes	Attributes	Instances
Min	2	3	185
Max	18	71	494051

Prototypical implementation

AMLBID

- 400 CASH scenarios from I4.0 AI domains
- 41 meta-features

Meta-features

Simple, Statistical & Information Theoretic their purpose is to measure the complexity of the underlying problem.

Model based measures are calculated by inducing a decision tree model on a dataset to get information about the hidden structures of the data.

Landmarking based measures that characterize the predictive problems when basic ML algorithms are performed on them.

Complexity based measures that analyze the complexity of a problem considering the overlap in the attributes values, the separability of the classes, and topological properties.

Prototypical implementation

AMLBID

- 400 CASH scenarios from I4.0 AI domains
- 41 meta-features
- 08 target algorithms and their configuration space

ML algorithms

- Support Vector Machines (C, Kernel, coef0, gamma, degree)
- Logistic Regression (C, penalty, fit_intercept)
- Decision Tree (max_features, min_samples_leaf, min_samples_split, criterion)
- Random Forest (bootstrap, max_features, min_samples_leaf /_split, split_criterion)
- Extra Trees (bootstrap, max_features, min_samples_leaf /_split, split_criterion)
- Gradient Boosting (learning_rate, n_estimators, depth, min_samples_leaf /_split)
- AdaBoost (algorithm, n_estimators, learning_rate, max_depth)
- Stochastic Gradient Descent (loss, penalty, learning_rate, l1 ratio, eta0, Power_t)

Prototypical implementation

AMLBID

- 400 CASH scenarios from I4.0 AI domains
- 41 meta-features
- 08 target algorithms and their configuration space
- +1000 Hyperparameters configuration

Pipelines generation

- 1000 HPs configurations for every algorithm $\mathcal{A}$ over each dataset $\mathcal{D}$
- 8000 pipelines for each dataset
- 10 x 5-fold stratified cross-validation strategy

Prototypical implementation

AMLBID

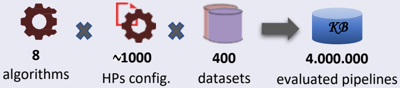
- 400 CASH scenarios from I4.0 AI domains
- 41 meta-features
- 08 target algorithms and their configuration space
- +1000 Hyperparameters configuration
- 4.000.000 evaluated pipelines in the KB

Pipelines generation

- 1000 HPs configurations for every algorithm $\mathcal{A}$ over each dataset $\mathcal{D}$
- 8000 pipelines for each dataset
- 10 x 5-fold stratified cross-validation strategy

Knowledge base

$$\mathcal{K}_B = \{(m_1, A^{(1)}_{H^1}), \ldots, (m_{400}, A^{(n)}_{H^{1000}})\}$$



Prototypical implementation

The Meta-model

Recommend the top-performing classification configurations for a combination of an unseen dataset and a classification evaluation measure

which?
- Random Forest
- k-Nearest Neighbor (kNN)

Why?
- of classification type
- sensitive
- can handle missing values
- extensible

Prototypical implementation

The Meta-model

Recommend the top-performing classification configurations for a combination of an unseen dataset and a classification evaluation measure

which?
- Random Forest
- **k-Nearest Neighbor (kNN)** ✔

Why?
- of classification type
- sensitive
- can handle missing values
- extensible

Empirical study

The experimental configuration

Benchmark datasets

- 30 datasets (binary and multiclass classification)
  - OpenML AutoML benchmark [Feurer *et al.* (2020)]
  - State-of-the-art papers [Garouani *et al.* (2022b)]

Baseline AutoML tools

- TPOT
  - Default settings (generation and evaluation of 100 pipelines for each dataset)
- Auto-sklearn
  - Auto-sklearn(V) : Vanilla version (Bayesian optimization)
  - Auto-sklearn(E) : Auto-sklearn 2.0 (Ensemble learning)

Empirical study

Experimental results : The recommendations performance

Table 1: Comparative performance analysis of AMLBID and the baseline AutoML tools.

Dataset	AMLBID	TPOT	Auto-sklearn(V)	Auto-sklearn(E)	Original paper result
[137]	**0.9374**	0.9120	0.8215	0.9283	0.8500
[138]	**0.9706**	0.9517	0.9632	0.9356	0.9500
[139]	**0.9941**	0.9907	0.9782	0.9900	0.9895
[141]	0.9205	**0.9991**	0.9357	0.6863	0.9984
[142]	0.8971	0.6711	0.9080	**0.9723**	0.9677
[143]	**0.9706**	0.7767	0.6780	0.9843	0.9278
[144]	**0.8967**	0.8899	0.6783	0.7952	0.8840
[145]	**0.8748**	0.7826	0.6702	0.7727	0.8659
Wafer-ds	**0.8571**	0.7312	0.8033	0.8953	-
vehicle	0.8880	0.8415	**0.9027**	0.6591	-
Cnae-9	**0.9671**	0.8803	0.7922	0.8365	-
Gas_Sens	0.9739	**0.9843**	0.9256	0.9468	-
Covertype	**0.8344**	0.7307	0.7890	0.6521	-
Kc1	**0.8793**	0.7097	0.7697	0.8552	-
⋮	⋱	⋱	⋱	⋱	⋮
jannis	0.6719	**0.7229**	0.6171	0.6845	-
MiniBooNE	**0.9645**	0.9423	0.8343	0.8903	-
Higgs	0.713	0.726	0.7135	**0.729**	-
Credi-g	**0.7921**	0.7188	0.5739	0.6121	-
kr-vs-kp	**0.9976**	0.9209	0.6532	0.7593	-
car	0.9754	**0.9999**	0.8549	0.9462	-
albert	**0.8759**	0.8005	0.8288	0.7981	-
airlines	0.6982	0.6758	**0.7094**	0.5927	-
Best performance	**19**	**6**	**2**	**3**	**-**

Empirical study

Experimental results : The recommendations performance

Table 1: Comparative performance analysis of AMLBID and the baseline AutoML tools.

Dataset	AMLBID	TPOT	Auto-sklearn(V)	Auto-sklearn(E)	Original paper result
[137]	**0.9374**	0.9120	0.8215	0.9283	0.8500
[138]	**0.9706**	0.9517	0.9632	0.9356	0.9500
[139]	**0.9941**	0.9907	0.9782	0.9900	0.9895
[141]	0.9205	**0.9991**	0.9357	0.6863	0.9984
[142]	0.8971	0.6711	0.9080	**0.9723**	0.9677
[143]	**0.9706**	0.7767	0.6780	0.9843	0.9278
[144]	**0.8967**	0.8899	0.6783	0.7952	0.8840
[145]	**0.8748**	0.7826	0.6702	0.7727	0.8659
Wafer-ds	**0.8571**	0.7312	0.8033	0.8953	-
vehicle	0.8880	0.8415	**0.9027**	0.6591	-
Cnae-9	**0.9671**	0.8803	0.7922	0.8365	-
Gas_Sens	0.9739	**0.9843**	0.9256	0.9468	-
Covertype	**0.8344**	0.7307	0.7890	0.6521	-
Kc1	**0.8793**	0.7097	0.7697	0.8552	-
⋮	⋱	⋱	⋱	⋱	⋮
jannis	0.6719	**0.7229**	0.6171	0.6845	-
MiniBooNE	**0.9645**	0.9423	0.8343	0.8903	-
Higgs	0.713	0.726	0.7135	**0.729**	-
Credi-g	**0.7921**	0.7188	0.5739	0.6121	-
kr-vs-kp	**0.9976**	0.9209	0.6532	0.7593	-
car	0.9754	**0.9999**	0.8549	0.9462	-
albert	**0.8759**	0.8005	0.8288	0.7981	-
airlines	0.6982	0.6758	**0.7094**	0.5927	-
Best performance	**19**	**6**	**2**	**3**	**-**

Empirical study

Experimental results : The recommendations performance

Table 1: Comparative performance analysis of AMLBID and the baseline AutoML tools.

Dataset	AMLBID	TPOT	Auto-sklearn(V)	Auto-sklearn(E)	Original paper result
[137]	**0.9374**	0.9120	0.8215	0.9283	0.8500 **(8.74)** ▲
[138]	**0.9706**	0.9517	0.9632	0.9356	0.9500 **(2.06)** ▲
[139]	**0.9941**	0.9907	0.9782	0.9900	0.9895 **(0.46)** ▲
[141]	0.9205	**0.9991**	0.9357	0.6863	0.9984 **(0.07)** ▲
[142]	0.8971	0.6711	0.9080	**0.9723**	0.9677 **(0.46)** ▲
[143]	**0.9706**	0.7767	0.6780	0.9843	0.9278 **(4.28)** ▲
[144]	**0.8967**	0.8899	0.6783	0.7952	0.8840 **(1.27)** ▲
[145]	**0.8748**	0.7826	0.6702	0.7727	0.8659 **(0.89)** ▲
Wafer-ds	**0.8571**	0.7312	0.8033	0.8953	-
vehicle	0.8880	0.8415	**0.9027**	0.6591	-
Cnae-9	**0.9671**	0.8803	0.7922	0.8365	-
Gas_Sens	0.9739	**0.9843**	0.9256	0.9468	-
Covertype	**0.8344**	0.7307	0.7890	0.6521	-
Kc1	**0.8793**	0.7097	0.7697	0.8552	-
⋮	⋱	⋱	⋱	⋱	⋮
jannis	0.6719	**0.7229**	0.6171	0.6845	-
MiniBooNE	**0.9645**	0.9423	0.8343	0.8903	-
Higgs	0.713	0.726	0.7135	**0.729**	-
Credi-g	**0.7921**	0.7188	0.5739	0.6121	-
kr-vs-kp	**0.9976**	0.9209	0.6532	0.7593	-
car	0.9754	**0.9999**	0.8549	0.9462	-
albert	**0.8759**	0.8005	0.8288	0.7981	-
airlines	0.6982	0.6758	**0.7094**	0.5927	-
Best performance	**19**	**6**	**2**	**3**	**-**

Empirical study

Experimental results : The run-time

Table 2: The run-time of the AMLBID, Autosklearn and TPOT tools on the benchmark datasets.

Dataset	Dataset size	AMLBID	Autosklearn	TPOT
[137]	959	**00:00:05**	01:23:47	00:08:14
[138]	2000	**00:00:12**	01:49:21	00:13:57
[139]	61000	**00:05:29**	04:19:05	03:42:09
[141]	274627	**00:11:43**	08:19:37	06:09:51
[142]	5000	**00:01:27**	02:31:07	01:38:36
[143]	1567	**00:00:53**	01:33:45	00:19:47
[144]	5388	**00:00:57**	01:56:50	00:55:51
[145]	1567	**00:00:33**	00:58:50	00:21:12
Wafer-ds	7306	**00:02:17**	03:44:26	01:42:21
vehicle	8463	**00:02:28**	02:12:40	01:45:40
Cnae-9	63260	**00:05:47**	04:07:39	03:24:52
Gas_Sens	4188	**00:01:14**	02:47:20	00:42:36
Covertype	25524	**00:03:04**	01:28:31	01:36:14
Kc1	2108	**00:00:38**	04:19:26	04:51:02
⋮	⋱	⋱	⋱	⋱
jannis	8641	**00:01:41**	02:31:07	01:41:51
MiniBooNE	52147	**00:04:23**	03:59:56	02:11:01
Higgs	110000	**00:06:16**	07:37:55	05:43:24
Credi-g	30000	**00:04:39**	02:03:34	05:33:03
kr-vs-kp	3196	**00:00:54**	01:17:19	00:22:44
car	1728	**00:00:38**	01:38:30	00:40:07
albert	43824	**00:06:27**	04:09:17	03:01:03
airlines	5473	**00:01:40**	02:18:27	00:57:52

Empirical study

Experimental results : The run-time

Table 2: The run-time of the AMLBID, Autosklearn and TPOT tools on the benchmark datasets.

Dataset	Dataset size	AMLBID	Autosklearn	TPOT
[137]	959	**00:00:05**	01:23:47	00:08:14
[138]	2000	**00:00:12**	01:49:21	00:13:57
[139]	61000	**00:05:29**	04:19:05	03:42:09
[141]	274627	**00:11:43**	08:19:37	06:09:51
[142]	5000	**00:01:27**	02:31:07	01:38:36
[143]	1567	**00:00:53**	01:33:45	00:19:47
[144]	5388	**00:00:57**	01:56:50	00:55:51
[145]	1567	**00:00:33**	00:58:50	00:21:12
Wafer-ds	7306	**00:02:17**	03:44:26	01:42:21
vehicle	8463	**00:02:28**	02:12:40	01:45:40
Cnae-9	63260	**00:05:47**	04:07:39	03:24:52
Gas_Sens	4188	**00:01:14**	02:47:20	00:42:36
Covertype	25524	**00:03:04**	01:28:31	01:36:14
Kc1	2108	**00:00:38**	04:19:26	04:51:02
⋮	⋱	⋱	⋱	⋱
jannis	8641	**00:01:41**	02:31:07	01:41:51
MiniBooNE	52147	**00:04:23**	03:59:56	02:11:01
Higgs	110000	**00:06:16**	07:37:55	05:43:24
Credi-g	30000	**00:04:39**	02:03:34	05:33:03
kr-vs-kp	3196	**00:00:54**	01:17:19	00:22:44
car	1728	**00:00:38**	01:38:30	00:40:07
albert	43824	**00:06:27**	04:09:17	03:01:03
airlines	5473	**00:01:40**	02:18:27	00:57:52

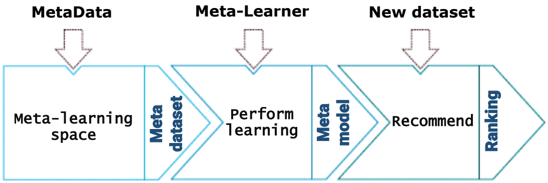
1 Context

2 Problem Statement and the State of the art

3 Research work
- Towards a Meta-learning based AutoML framework for Industrial big data
- **Learning abstract tasks representation**
- Towards interactive explainable AutoML
- AMLBID : a self-explainable AutoML software package

4 Conclusion & perspectives

Meta-learning



- Appropriate data characterization is crucial for the meta-learning
- Proper form of data characterization can guide the process of learning algorithms selection and configuration

Data characterization

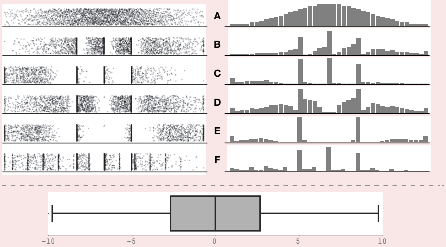
Hand-designed meta-features

- Simple, Statistical & Info. theoretic
- Landmarking
- Model-based
- Data Complexity

But?

What criteria should we invoke to include or discard a family of meta-features?

Datasets may share identical statistical properties but noticeably they have different data distributions.

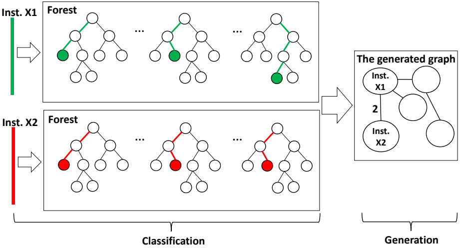


[Matejka *et al.* (2017)]

Data characterization

Graph-based dataset Representation

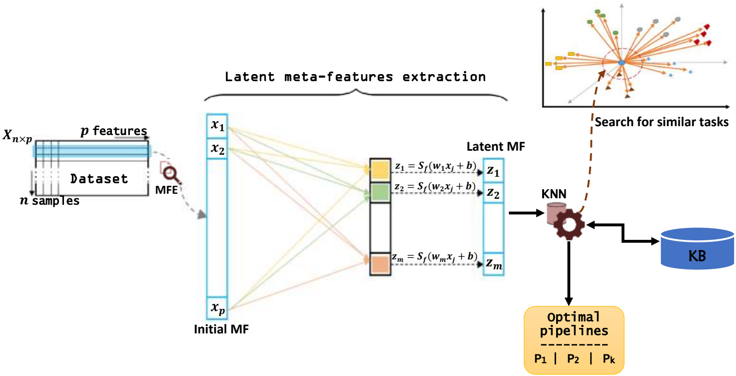
- Represents datasets as graphs and then extracts their latent representation.
- Vertices represent the dataset instances
- Edges indicate the existence of a sufficiently high co-occurrence score among them.



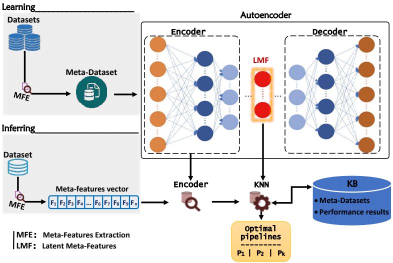
[Cohen-Shapira *et al.* (2019)]

This approach suffers from a computational complexity of $O(V^4)$ where V is the number of vertices in the analyzed graph.

The AeKNN meta-model with built in data characterization



The AeKNN meta-model

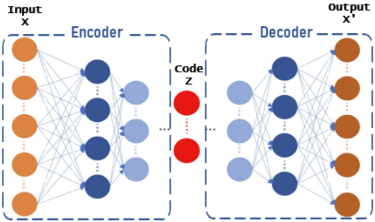


Algorithm: AeKNN algorithm's pseudo-code.

Input: Train Data, Test Data, KB ▷ KB is the constructed knowledge base
Output: P$< P_1, P_2, P_3, \ldots, P_n >$ ▷ Suggested pipelines
Learning phase:
$MetaData \leftarrow$ `MetaFeaturesExtractor(`$TrainData$`)`
$AE \leftarrow$ `Autoencoder(`$MetaData$`)`
$EncoderModel \leftarrow$ `FeedForwardAEModel(`AE`)`
$LatentMetaFeatures \leftarrow$ `EncoderModel(`$TrainData$`)`
$AeKNN \leftarrow$ `KNN(`$LatentMetaFeatures$`, KB)`
Inferring phase:
$MetaFeatures \leftarrow$ `MetaFeaturesExtractor(`$TestData$`)`
$LatentMetaFeatures \leftarrow$ `EncoderModel(`$MetaFeatures$`)`
$OptimalPiplines \leftarrow$ `AeKNN(`$LatentMetaFeatures$`, `KB`)`

AekNN foundations

Autoencoders



Encoder

$Z = E(X)$ that encodes the high dimensional input data $X = \{x_1, x_2, .., x_n\}$ into a low dimensional hidden representation $Z = \{z_1, z_2, z_m\}$ by an activation function f

Decoder

decoding function $X' = D(Z)$ that produces a reconstruction of the inputs $X' = \{x'_1, x'_2, \ldots, x'_n\}$, while minimizing the reconstruction error $L(X, X')$.

$$L(X, X') = -\sum_{i=1}^{n}(x_i \log x'_i) + (1 - x_i)(x_i \log(1 - x'_i))$$

Experimental study

AeKNN architectures analysis

AeKNN is characterized by the l_i^n parameter that establishes the architecture of the network. This parameter allows the selection of different architectures in terms of depth (number of layers) and number of neurons per layer.

Table 3: Experimental configurations of AeKNN.

Model	Number of hidden layers	Number of neurons per layer					Architecture l_i^n
		L 1	L 2	Latent layer	L 4	L 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 AeKNN meta-model

AeKNN architectures analysis

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
Best performance	**14**	**3**	**0**	**2**	**1**

The AeKNN meta-model

AeKNN architectures analysis

Table 4: **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
Best performance	**8**	**5**	**5**	**1**	**1**

The AeKNN meta-model

AeKNN architectures analysis

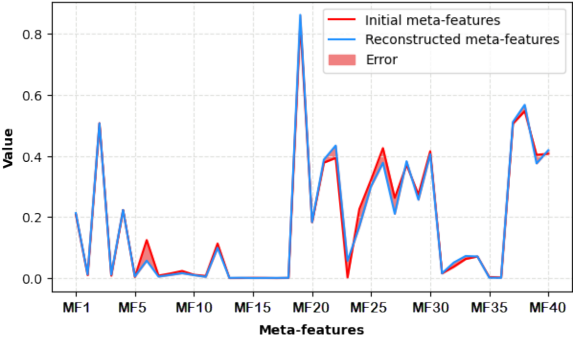
Table 4: **AUC** classification results of the recommended pipelines for the considered AeKNN architectures.

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
Best performance	**11**	**6**	**3**	**0**	**0**

The AeKNN meta-model

AeKNN architectures analysis

It is considered that $l_i{}^n = (32)$ is the best among the considered architectures with a reconstruction error standard deviation of 0.020025



The AeKNN meta-model

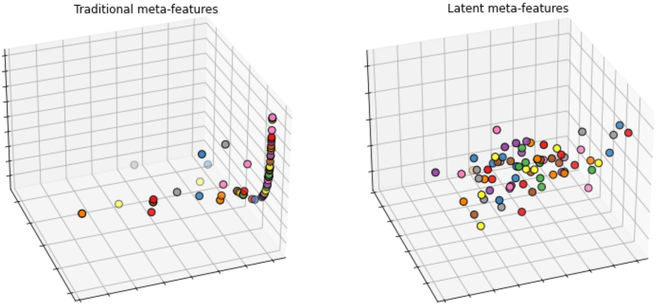
Results of the algorithms selection process

Table 5: Results of RF, XGB, KNN, and AeKNN meta-models for recommending optimal pipelines for test data.

Dataset	Accuracy			
	AeKNN	KNN	XGB	RF
APSFailure	**0.9921** (0.11) ▲	0.9910	0.9673	0.8950
Higgs	**0.7283** (1.53) ▲	0.7130	0.6801	0.6072
CustSat	0.8155 (4.04) ▼	0.8559	**0.8715**	0.7382
car	**0.9999** (2.45) ▲	0.9754	0.9462	0.8549
kr-vs-kp	**0.9985** (0.09) ▲	0.9976	0.7593	0.6532
airlines	0.7021 (0.39) ▲	0.6982	**0.7094**	0.5927
vehicle	0.8952 (0.72) ▲	0.8880	**0.9027**	0.6591
MiniBooNE	**0.9730** (0.85) ▲	0.9645	0.8903	0.8343
jannis	**0.7229** (5.10) ▲	0.6719	0.6845	0.6171
nomao	**0.9884** (1.76) ▲	0.9708	0.7987	0.6995
Credi-g	**0.8037** (1.16) ▲	0.7921	0.5739	0.6121
Kc1	**0.8905** (1.12) ▲	0.8793	0.7697	0.7097
Cnae-9	**0.9800** (1.29) ▲	0.9671	0.8365	0.7922
albert	**0.8790** (0.31) ▲	0.8759	0.8288	0.7981
Numerai28.6	**0.5591** (3.84) ▲	0.5207	0.4836	0.4229
segment	**0.9867** (1.32) ▲	0.9735	0.9542	0.9337
Covertype	**0.8637** (2.93) ▲	0.8344	0.7890	0.6521
KDDCup	**0.9781** (0.41) ▲	0.9740	0.9331	0.8934
shuttle	0.9362 (2.87) ▼	**0.9649**	0.9649	0.8429
Gas_Sens-uci	**0.9843** (1.04) ▲	0.9739	0.9468	0.9256

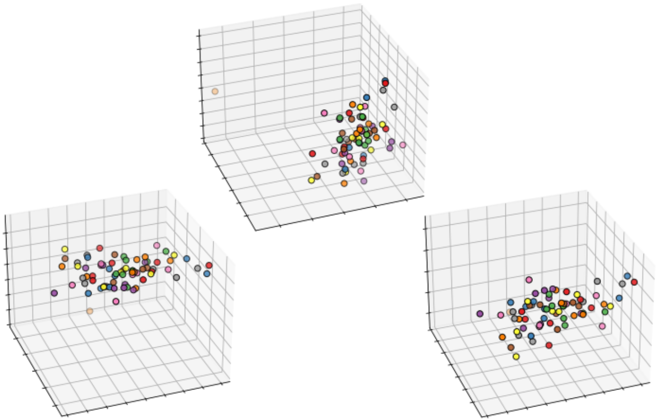
The AeKNN meta-model

Results of latent meta-features extraction



The AeKNN meta-model

Results of latent meta-features extraction



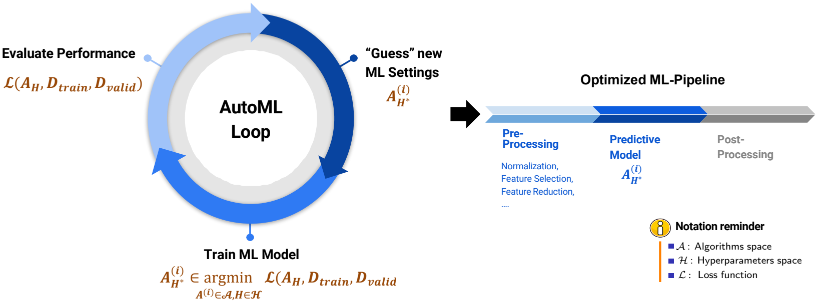
1 Context

2 Problem Statement and the State of the art

3 Research work
- Towards a Meta-learning based AutoML framework for Industrial big data
- Learning abstract tasks representation
- Towards interactive explainable AutoML
- AMLBID : a self-explainable AutoML software package

4 Conclusion & perspectives

AutoML Process



Notation reminder
- $\mathcal{A}$: Algorithms space
- $\mathcal{H}$: Hyperparameters space
- $\mathcal{L}$: Loss function

Fully automated ML design can also receive pushback
- 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?
- How to verify results?

Humans and AutoML

Who is using AutoML?

Users without any deep expertise in ML

ML experts & researchers, data scientists

[Bouthillier *et al.* (2020)] showed that authors of NeurIPS and ICLR papers :

- often **optimize their pipelines hyperparameters** ($> 75\%$)
- often **do it manually** and don't use AutoML tools

[Crisan *et al.* (2021)] interviewed data scientists and concluded :

- experts **don't necessarily trust** AutoML
- **visualization** of results and **interaction** with processes can help to increase the acceptance of AutoML

Towards Interactive eXplainable AutoML (IXAutoML)

What we are aiming for?

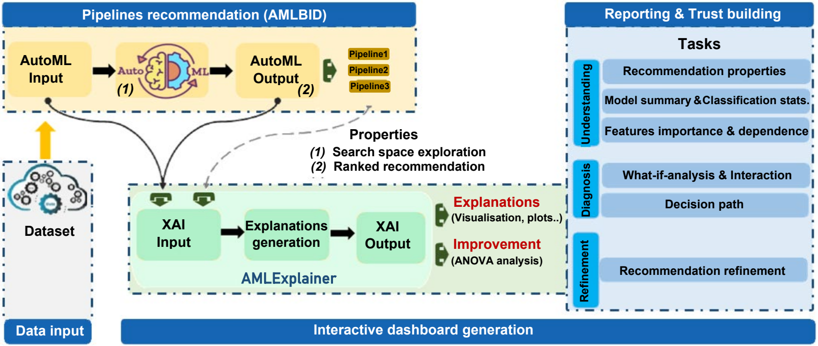
An ideal XAI system should be flexible enough to adapt to the AutoML output (model and data agnostic).

Interpretability
How a prediction is made by the model

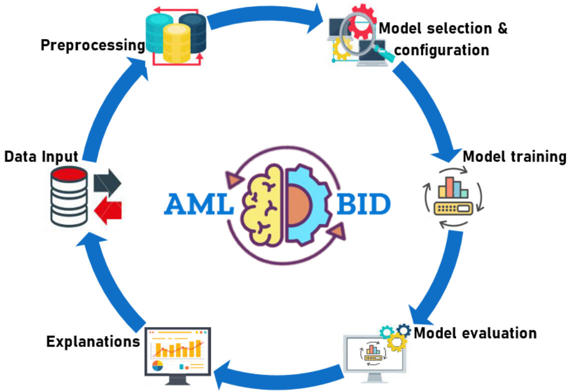
Explainability
Why can we learn from the model

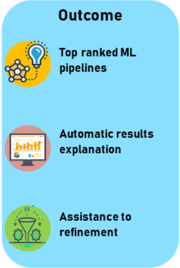
Trustworthiness
How trustworthy is the model's prediction

Towards Interactive eXplainable AutoML (IXAutoML)



Demonstration





1 Context

2 Problem Statement and the State of the art

3 Research work
- Towards a Meta-learning based AutoML framework for Industrial big data
- Learning abstract tasks representation
- Towards interactive explainable AutoML
- AMLBID : a self-explainable AutoML software package

4 Conclusion & perspectives

AMLBID : Democratization of explainable machine learning

- It is open-source (MIT) and trivial to use.

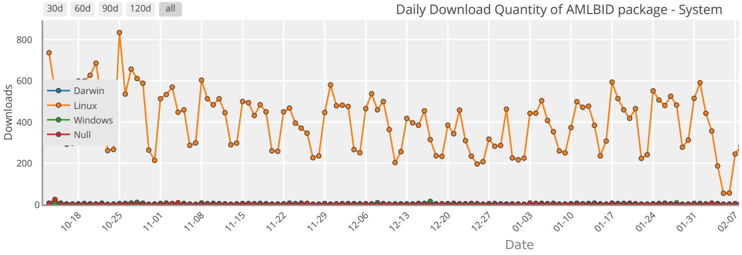
```
from AMLBID.recommender import AMLBID_Recommender
from AMLBID.explainer import AMLBID_Explainer

model,config=AMLBID_Recommender.recommend(Data, metric, mode)
model.fit(X_train, Y_train)

Explainer = AMLBID_Explainer.explain(model, config, Data)
Explainer.dash()
```

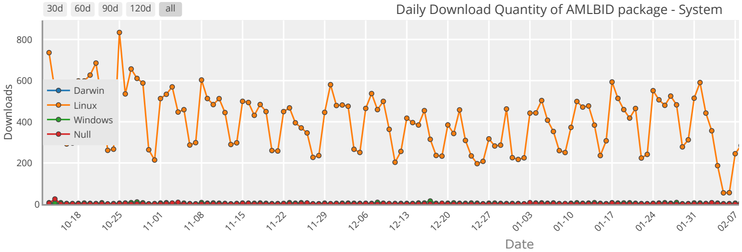
AMLBID : Democratization of explainable machine learning

- It is open-source (MIT) and trivial to use.
- Downloaded more than **17.753** times on PyPI in its first year.



AMLBID : Democratization of explainable machine learning

- It is open-source (MIT) and trivial to use.
- Downloaded more than **17.753** times on PyPI in its first year.
- Multiple industrial requests.



1 Context

2 Problem Statement and the State of the art

3 Research work
- Towards a Meta-learning based AutoML framework for Industrial big data
- Learning abstract tasks representation
- Towards interactive explainable AutoML
- AMLBID : a self-explainable AutoML software package

4 Conclusion & perspectives

Perspectives

Expand AMLBID

- Support the algorithms of :
  - Regression
  - Deep learning
  - Distributed ML (Spark ML)
- Cover the tasks of :
  - Data pre-processing
  - Features engineering
  - Post-processing analysis
- Enrich the Meta-KB from collaborative ML platforms (Kaggle, OpenML, etc.)
- Explore the inclusion of AutoXAI in the `AMLexplainer` explanatory artefact
- Explore the use of the constructed knowledge base for further guidance and automation of ML applications