Interpretable Automated Machine Learning in Maana™ Knowledge Platform

Extended Abstract

Alexander Elkholy, Fangkai Yang, Steven Gustafson
Maana, Inc.
Bellevue, Wa
aelkholy@maana.io

KEYWORDS

AutoML; Machine learning; Knowledge representation

ACM Reference Format:
Alexander Elkholy, Fangkai Yang, Steven Gustafson. 2019. Interpretable Automated Machine Learning in Maana™ Knowledge Platform . In *Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019*, IFAAMAS, 3 pages.

Machine learning is an essential part of developing solutions for many industrial applications. Developers of such applications need to rapidly build, test, deploy and validate machine learning models. While the validation of models is a key capability that will enable industries to more widely adopt machine learning capabilities for business decision making, this process suffers from the lack of *interpretability*. Validation usually begins with understanding how a machine learning model is developed - the pipeline from source data, through data processing and featurization, to model building and parameter tuning. The ability to understand the machine learning model also represents a key piece of domain knowledge: a data scientist who understands how to make successful models will be in high demand across that domain because they understand how to combine raw data with machine learning tools and produce valuable outcomes. Unfortunately, this level of understanding remains somewhat of a "dark art" in that the knowledge and judgment used to find good domain-specific machine learning pipelines is usually found in the heads of the data scientists with domain knowledge and developed over time through experience. Therefore, while it is possible to see the final machine learning pipeline and code implementing it, the steps that the data scientist went through, and the compromises and decisions they made, are not captured. Once the model is delivered, most insights and assumptions related to development of the solution are lost, making long term sustaining of the model development operation difficult. There is also no general way of tracking accumulated experiences in building a given model often useful in building similar solutions.

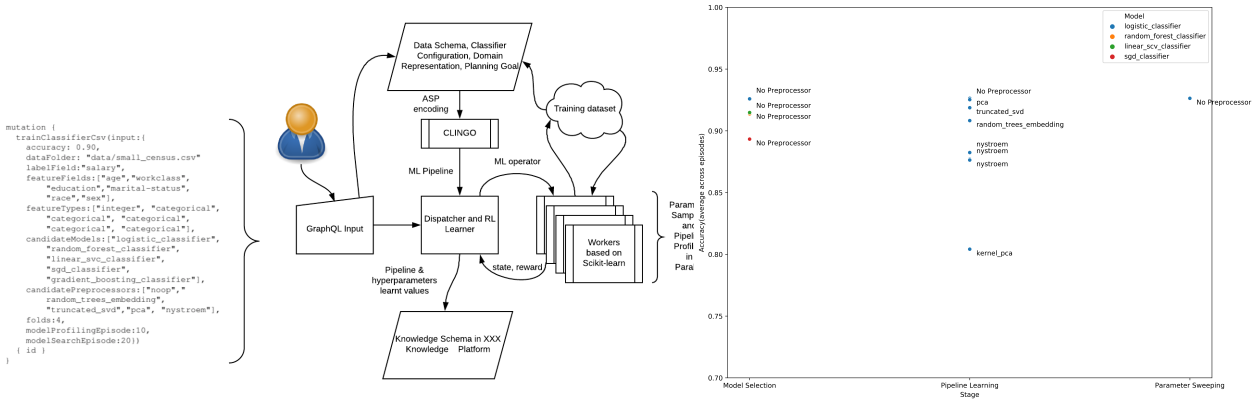
To facilitate rapid development and deployment of data science solutions, automated machine learning (AutoML) has gained more interest recently due to the availability of public dataset repositories and open source machine learning code bases. AutoML systems such as Auto-WEKA [10], Auto-SKLEARN [2] and TPOT[6] attempt to optimize the entire machine learning pipeline. These pipelines consist of independent steps such as featurization to encode data in numeric form and feature selection to pick the best subset of features. Given sufficient computation resources, these system can achieve good accuracy in building machine learning pipelines, but they do not provide clear explanations to justify the choice of models that can be verified by data scientist, and consequently the problem of interpretability remains unsolved.

To address this challenge, we describe the **Maana Meta-learning** service which provides *interpretable automated machine learning*. The goal of this project is two-folded. Firstly, we hope that the efficiency of developing data science solutions can be improved by leveraging an automated search and profiling algorithm such that a baseline solution can be automatically generated for the data scientists to fine-tune. Secondly, we hope that such automated search process is transparent to human users, and through learning process the service can return interpretable insights on the choice of models and hyper-parameters and encode them as knowledge. Contrasted with most AutoML systems that provide end-to-end solutions, the Maana Meta-learning service is an interactive assistant to data scientists that performs *user-guided, machine-assisted* automated machine learning. By having data scientists specify a pre-determined search space, and Meta-learning service then goes through several stages to perform model selection, pipeline profiling and hyper-parameter tuning. During this process, it returns intermediate results and user can inject feedback to steer the search process. Finally, it generates an optimal pipeline along with structured knowledge encoding the decision making process, leading to an interpretable automated machine learning process.

Maana Meta-Learning service features two components: (1) a knowledge representation that captures domain knowledge of data scientists and (2) an AutoML algorithm that generates machine learning pipeline, evaluates their efficacy by sampling hyper-parameters, and encodes all the information about the choices made and subsequent performance / parameters into the knowledge representation. The knowledge representation is defined using GraphQL[1]. Developed by Facebook as an alternative to the popular REST interface [3], GraphQL provides only a single API endpoint for data access, backed by a structured, hierarchical type system. Consequently, it allows us to define a knowledge taxonomy to capture concepts of machine learning pipelines, seamlessly populate facts to the predefined knowledge graph and reason with them. The AutoML algorithm, in charge of generating and choosing which pipelines to pursue, is based on PEORL framework [11], an integration of symbolic planning [1] and hierarchical reinforcement

[1] https://graphql.org/

Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), N. Agmon, M. E. Taylor, E. Elkind, M. Veloso (eds.), May 13–17, 2019, Montreal, Canada.



(a) The Meta-learning Service Architecture.

(b) Choices made in each of the three stages for an example run.

Figure 1: Meta-learning Service Overview

Dataset	Pipeline	CV accuracy
Reuters 50/50	none/hashing vectorizer/linear SVC	0.848
IMDB	none/hashing vectorizer/SGD classifier	0.879
Adult	random trees/one hot & min_max scaling/SGD	0.8523
Spam detection	None/min_max scaling/logistic regression	0.927
Parkinsons detection	None/std_scaler/random forest	0.887
Abalone	Nystroem/std_scaler/random forest	0.552
Car	Nystroem/one hot/gradient boosting	0.938

Table 1: Baseline Pipelines Learned on Datasets

learning [9]. Symbolic plans generated from a pre-defined symbolic formulation of a dynamic domain is used to guide reinforcement learning, and recently this approach is generalized to improve interpretability of deep reinforcement learning. In the setting of AutoML, generating machine learning pipelines is treated as a symbolic planning problem on an action description in action language $\mathcal{BC}$[4] that contains actions such as *preprocessing*, *featurizing*, *cross validation*, *training* and *prediction*. The architecture of the system is shown in Figure 1a.

The pipeline is sent to execution where each symbolic action by mapping to primitive actions in a Markov Decision Process [7] (MDP) space, which are ML pipeline components instantiated with random hyper-parameters, in order to learn the quality of the actions in the pipeline. The learning process is value iteration on R-learning [5, 8], where cross-validation accuracy of the pipeline is used as rewards. After the quality of the current pipeline is measured, an improved ML pipeline is generated thereafter using the learned values, and the interaction with learning continues, until no better pipeline can be found. This step is called *model profiling*. After that, a more systematic parameter sweeping is performed, i.e., *model searching*. This allows us to describe the pipeline steps in an intuitive representation and explore the program space more systematically and efficiently with the help of reinforcement learning.

One can see the progress as the algorithm moves through the different search stages on the Spam detection dataset, settling on the logistic regression classifier in Figure 1b.

In Maana Knowledge Platform, CSV files can be uploaded and each column becomes a field and their types are automatically identified. The user can trigger Meta-learning service by submitting a query through GraphQL endpoint. The GraphQL input is used to generate part of the initial state for planning, and the Meta-learning service is triggered for pipeline search. Throughout the pipeline search process, the results are constantly written to the Maana Knowledge platform according to the knowledge schema. The service is implemented in Python with Graphene library to enable GraphQL server and endpoints. It is deployed using a Docker image along with other components of Maana Knowledge platform.

Additionally, another feature we use to improve performance is the parallelization of building models. Because the model profiling and model search episodes can be done in parallel, we use asynchronous approach, where multiple workers are launched and each perform their own parameter sampling and cross validation on the dataset, and the result is returned to the dispatcher to perform value iteration.

The results in Table 1 show that the Meta-learning service generates competitive baseline result for the data scientist to further work on with standard datasets from the UCI Machine Learning Repository[2]. A strongly-typed representation of results via GraphQL, combined with the $\mathcal{BC}$ action language configure pipelines and rules and an effective search using PEORL, produce a transparent AutoML platform.

[2]https://archive.ics.uci.edu/ml/index.php

REFERENCES

[1] A. Cimatti, M. Pistore, and P. Traverso. 2008. Automated planning. In *Handbook of Knowledge Representation*, Frank van Harmelen, Vladimir Lifschitz, and Bruce Porter (Eds.). Elsevier.

[2] Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. 2015. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems*. 2962–2970.

[3] Roy T Fielding and Richard N Taylor. 2000. *Architectural styles and the design of network-based software architectures*. Vol. 7. University of California, Irvine Doctoral dissertation.

[4] Joohyung Lee, Vladimir Lifschitz, and Fangkai Yang. 2013. Action Language $\mathcal{BC}$: A Preliminary Report. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

[5] S. Mahadevan. 1996. Average Reward Reinforcement Learning: Foundations, Algorithms, and Empirical Results. *Machine Learning* 22 (1996), 159–195.

[6] Randal S Olson, Nathan Bartley, Ryan J Urbanowicz, and Jason H Moore. 2016. Evaluation of a tree-based pipeline optimization tool for automating data science. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*. ACM, 485–492.

[7] M. L. Puterman. 1994. *Markov Decision Processes*. Wiley Interscience, New York, USA.

[8] A. Schwartz. 1993. A Reinforcement Learning Method for Maximizing Undiscounted Rewards. In *Proc. 10th International Conf. on Machine Learning*. Morgan Kaufmann, San Francisco, CA.

[9] R. Sutton and A. G. Barto. 1998. *Reinforcement Learning: An Introduction*. MIT Press.

[10] C. Thornton, F. Hutter, H. H. Hoos, and K. Leyton-Brown. 2013. Auto-WEKA: Combined Selection and Hyperparameter Optimization of Classification Algorithms. In *Proc. of KDD-2013*. 847–855.

[11] F. Yang, D. Lyu, B. Liu, and S. Gustafson. 2018. PEORL: Integrating Symbolic Planning and Hierarchical Reinforcement Learning for Robust Decision-Making. In *International Joint Conference of Artificial Intelligence (IJCAI)*.