Gradual AutoML using Lale

Martin Hirzel
IBM Research
USA

Kiran Kate
IBM Research
USA

Parikshit Ram
IBM Research
USA

Avraham Shinnar
IBM Research
USA

Jason Tsay
IBM Research
USA

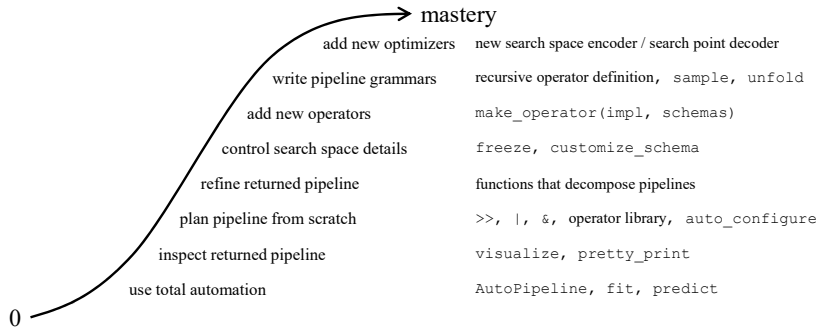

Figure 1: Gradual AutoML

ABSTRACT

Lale is a sklearn-compatible library for automated machine learning (AutoML). It is open-source (https://github.com/ibm/lale) and addresses the need for gradual automation of machine learning as opposed to offering a black-box AutoML tool. Black-box AutoML tools are difficult to customize and thus restrict data scientists in leveraging their knowledge and intuition in the automation process. Lale is built on three principles: progressive disclosure, orthogonality, and least surprise. These enable a gradual approach offering a spectrum of usage patterns starting from total automation to controlling almost every aspect of AutoML. Lale provides compositional constructs that let data scientists control some aspects of their pipelines while leaving other aspects free to be searched automatically. This tutorial demonstrates the use of Lale for various machine-learning tasks, showing how to progressively exercise more customization. It also covers AutoML for advanced scenarios such as class imbalance correction, bias detection and mitigation, multi-objective optimization, and working with multi-table datasets. While Lale comes with hyperparameter specifications for 216 operators out-of-the-box, users can also add more operators of their own, and this tutorial covers how to do that. Overall, this tutorial teaches you how you can exercise fine-grained control over AutoML without having to be an AutoML expert.

KDD '22, August 14–18, 2022, Washington, DC, USA

ACM ISBN 978-1-4503-9385-0/22/08.
https://doi.org/10.1145/3534678.3542630

KEYWORDS

Programming models, AutoML, datasets, automated data science.

ACM Reference Format:
Martin Hirzel, Kiran Kate, Parikshit Ram, Avraham Shinnar, and Jason Tsay. 2022. Gradual AutoML using Lale. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3534678.3542630

1 TARGET AUDIENCE AND PREREQUISITES

This tutorial targets data scientists who want to leverage AutoML. It expects some familiarity with Python libraries such as numpy, pandas, and sklearn [3] (our user study [1] showed that data scientists with moderate sklearn experience can successfully use Lale to solve advanced tasks). No AutoML experience is required.

2 INTRODUCTION

Automated machine learning (AutoML) searches over a set of operators and their hyperparameters. This search space can be predefined or can be customized based on application constraints as well as the experience of the data scientist. However, with most current AutoML tools, it is hard to exercise control over any of these choices. Lale is an open-source library that provides *gradual AutoML* to address the ease of use as users customize the AutoML process. Figure 1 shows how gradual AutoML progresses from total automation to total customization. Total automation works with a pre-defined ML pipeline and uses a default optimizer to search over the choices in that pipeline. Users can also inspect the output of automation, modify it, re-run automation with a custom search space, create new operators and search space specifications, create a pipeline topology search through pipeline grammars, and even customize or add a new optimization algorithm.

Lale also addresses other needs in model building and automation such as handling class imbalance, bias metrics and mitigation, multi-objective optimization, and working with multi-table datasets. This tutorial will illustrate these cases along with gradual AutoML.

3 TUTORIAL OUTLINE

The tutorial comprises multiple short sections, each addressing a task or a sub-task of AutoML. Most sections will use Jupyter notebooks for hands-on exercises of using Lale for that task.

3.1 Introduction to AutoML

This section will provide an overview of AutoML in general and introduce concepts such as hyperparameter optimization (HPO) and combined algorithm selection and hyperparameter tuning (CASH). We will also explain the concept of gradual AutoML [1].

3.2 Total Automation with Lale

Lale provides a simple sklearn-style operator called `AutoPipeline` to achieve total automation for standard machine learning tasks such as classification and regression on tabular data. We will demonstrate how to use `AutoPipeline` in just 3 lines of code, and how to inspect the output models of `AutoPipeline`.

3.3 Customizing Algorithm Choices and Hyperparameters

This section will focus on using Lale to compose a pipeline with algorithm choices, such as between different categorical encoders and between different classifiers. Users can then perform CASH on this pipeline using `auto_configure`. We will also demonstrate how to refine outputs of AutoML and do iterative AutoML.

3.4 Handling Class Imbalance

Lale includes operators for class imbalance correction from imbalanced-learn [5]. This section introduces the concept of higher-order operators and shows how to use them to create pipelines involving down-sampling and up-sampling.

3.5 Bias Mitigation

To the best of our knowledge, Lale is the first AutoML library that allows easy inclusion of bias mitigators in the search space. This section introduces fairness metrics and bias mitigators from AIF360 [2] and demonstrates CASH over them [4].

3.6 Multi-objective Optimization

Most commonly used AutoML tools optimize for a single metric. In practice, there is often a need to take more than one metric into account while searching for the best model. For example, we may want good predictive performance as well as model fairness. This section will demonstrate the use of Lale for such multi-objective optimization to find Pareto frontiers.

3.7 Working with Multi-table Datasets

The preprocessing steps in a data science workflow often involve more than one table. Data scientists usually write independent scripts for such data preparation as there is no easy way to include it in a machine learning pipeline. Lale introduced preprocessing operators for performing join, filter, map, groupby, aggregate, etc. as part of end-to-end sklearn-style pipelines [6]. We will demonstrate these operators in the context of a standard machine learning task.

3.8 Adding a New Operator

An operator in Lale is a light-weight wrapper over existing algorithmic implementations. For example, Lale has wrappers for 216 existing implementations from sklearn [3], imbalanced-learn [5], AIF360 [2], etc. Adding a new operator in Lale is relatively straightforward and well documented. This section walks through the steps to add a new operator wrapper and to define a search space for that operator's hyperparameters.

3.9 Lale Internals

This section will discuss how Lale employs compiler techniques to generate search spaces for different optimizer backends, for early error reporting, and for automatic documentation generation.

3.10 Research Directions

We will wrap up the tutorial with examples of some research directions: (1) batch-wise training of pipelines for datasets that do not fit in main memory, and (2) grammars to define recursive search spaces. We will cover examples of when and how these capabilities can be used.

4 CONCLUSION

We hope that Lale will contribute towards making AutoML easier to use and control by data scientist.

Acknowledgments

We would like to thank Guillaume Baudart (DI ENS, Ecole normale supérieure, PSL University, CNRS, INRIA, France), Michael Feffer (Carnegie Mellon University, USA), Louis Mandel (IBM Research, USA), Chirag Sahni (Rensselaer Polytechnic Institute, USA), and Vaibhav Saxena (IBM Research, India) for their valuable contributions to Lale which will be included in the tutorial.

REFERENCES

[1] Guillaume Baudart, Martin Hirzel, Kiran Kate, Parikshit Ram, Avraham Shinnar, and Jason Tsay. 2021. Pipeline Combinators for Gradual AutoML. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[2] Rachel K. E. Bellamy, Kuntal Dey, Michael Hind, Samuel C. Hoffman, Stephanie Houde, Kalapriya Kannan, Pranay Lohia, Jacquelyn Martino, Sameep Mehta, Aleksandra Mojsilovic, Seema Nagar, Karthikeyan Natesan Ramamurthy, John Richards, Diptikalyan Saha, Prasanna Sattigeri, Moninder Singh, Kush R. Varshney, and Yunfeng Zhang. 2018. AI Fairness 360: An Extensible Toolkit for Detecting, Understanding, and Mitigating Unwanted Algorithmic Bias. https://arxiv.org/abs/1810.01943

[3] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. 2013. API Design for Machine Learning Software: Experiences from the scikit-learn Project. https://arxiv.org/abs/1309.0238

[4] Martin Hirzel, Kiran Kate, and Parikshit Ram. 2021. Engineering Fair Machine Learning Pipelines. In *ICLR Workshop on Responsible AI (RAI@ICLR)*.

[5] Guillaume Lemaître, Fernando Nogueira, and Christos K. Aridas. 2017. Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning. *Journal of Machine Learning Research* 18, 17 (2017), 1–5.

[6] Chirag Sahni, Kiran Kate, Avraham Shinnar, Hoang Thanh Lam, and Martin Hirzel. 2021. RASL: Relational Algebra in Scikit-Learn Pipelines. In *Workshop on Databases and AI (DBAI@NeurIPS)*.