The Journal of Open Source Software

NiaAML: AutoML framework based on stochastic population-based nature-inspired algorithms

Luka Pečnik[*1] **and Iztok Fister Jr.**[1]

1 University of Maribor, Faculty of Electrical Engineering and Computer Science

DOI: 10.21105/joss.02949

Software

- Review
- Repository
- Archive

Editor: Arfon Smith

Reviewers:

- @adi3
- @sara-02

Submitted: 16 December 2020
Published: 23 May 2021

Summary

The field of Automated Machine Learning (AutoML) has been developed to automate data preprocessing and search for optimal algorithms together with their hyperparameters in order to discover the best possible ML pipeline for an input dataset (Hutter et al., 2019). AutoML can be modeled as a continuous optimization problem with several potential optimization methods considered. Stochastic population-based nature-inspired algorithms (Engelbrecht, 2007; Yang, 2014) are a popular class of tools for dealing with such continuous optimization problems. These algorithms are inspired mainly by the biological behavior of various species living in nature (Fister Jr et al., 2013). Such algorithms are composed of a population of individuals that undergo different variation operations during the evolution process which results in new populations. The Python framework we have developed, NiaAML, incorporates these stochastic algorithms to search for the most suitable classification pipeline in a dataset (Fister Jr. et al., 2020).

The framework is developed in a layer style layout architecture consisting of several components, including feature selection algorithms, feature transformation algorithms and classifiers. Its task is to find a perfect combination of components with proper classifier hyperparameter settings to build an efficient, yet customizable classification pipeline with the help of a popular collection of nature-inspired algorithms, named NiaPy (Vrbančič et al., 2018). NiaAML incorporates two types of optimizations, the first involves finding the optimal set of components for the pipeline, and the second involves tuning the hyperparameters. Users can freely choose the ML components to be included into the optimization process, as well as select suitable fitness functions to be used for evaluation of candidate pipelines. Input data can be in the form of numerical and categorical features, as well as missing attributes, while pipelines are exported and imported as binary files for post-hoc use. Further, they can be exported as user-friendly text files that contain all of the relevant information about the pipeline and its components. A graphical outline of the NiaAML method is presented in Figure 1.

*Corresponding author

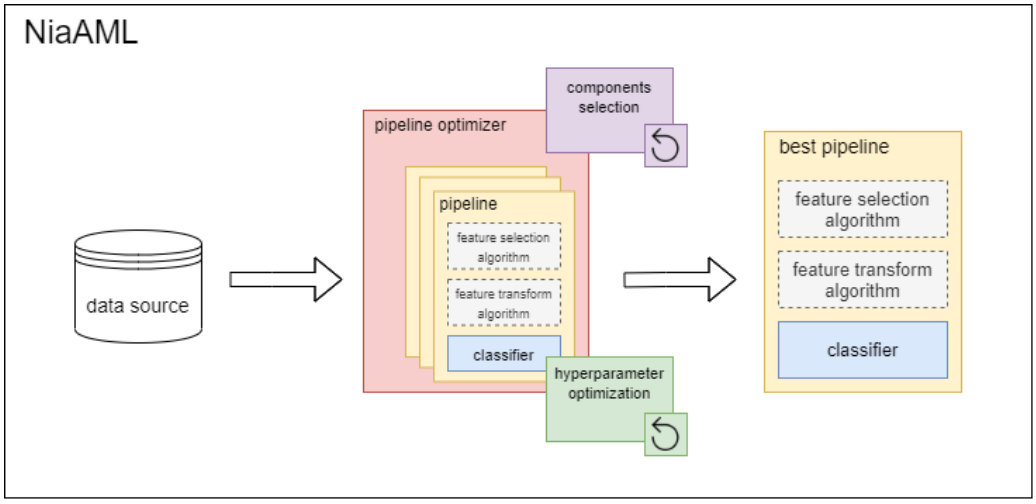

Figure 1: NiaAML flow.

Statement of need

Searching for an optimal classification pipeline in ML that provides the best results for a particular classification task involves a lot of domain-specific knowledge and numerous trial-and-error approaches. For instance, several conditions must be fulfilled to adequately apply an ML algorithm, such as proper preparation and preprocessing of input data, and selection of appropriate classifiers as well as their parameters. Due to this, mostly ML experts and data scientists have been able to handle such tasks in the past. However, empirical evidence shows that automation of the optimal classifier selection process, feature preprocessing steps and their hyperparameters, can be dealt with by non-experts as well (Feurer et al., 2015; Guyon et al., 2015).

With the rise of AutoML methods, dealing with ML has also become available to researchers from other fields. Compared to similar Python AutoML frameworks, such as TPOT (Olson & Moore, 2019) and auto-sklearn (Feurer et al., 2015), NiaAML offers the following benefits:

1. It is fully modeled as a continuous optimization problem, which means that arbitrary stochastic nature-inspired population-based algorithms can be used for solving this task without any special modifications of their internal mechanisms.
2. The search for the optimal combination of ML components and proper classifier hyper-parameters can be conducted concurrently.
3. Its layer style architecture allows the straightforward addition of new ML components.
4. Every pipeline in the output is feasible and functional in cases of the correct specified hyperparameters domains.
5. A Graphical User Interface (GUI) further simplifies the work for users without expert domain knowledge.

References

Engelbrecht, A. P. (2007). *Computational intelligence: An introduction*. John Wiley & Sons. https://doi.org/10.1002/9780470512517

Feurer, M., Klein, A., Eggensperger, K., Springenberg, J., Blum, M., & Hutter, F. (2015). Efficient and robust automated machine learning. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, & R. Garnett (Eds.), *Advances in neural information processing systems* (Vol. 28, pp. 2962–2970). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2015/file/11d0e6287202fced83f79975ec59a3a6-Paper.pdf

Fister Jr, I., Yang, X.-S., Fister, I., Brest, J., & Fister, D. (2013). A brief review of nature-inspired algorithms for optimization. *arXiv Preprint arXiv:1307.4186*.

Fister Jr., I., Zorman, M., Fister, D., & Fister, I. (2020). Continuous optimizers for automatic design and evaluation of classification pipelines. In *Frontier applications of nature inspired computation* (pp. 281–301). https://doi.org/10.1007/978-981-15-2133-1_13

Guyon, I., Bennett, K., Cawley, G., Escalante, H. J., Escalera, S., Tin Kam Ho, Macià, N., Ray, B., Saeed, M., Statnikov, A., & Viegas, E. (2015). Design of the 2015 ChaLearn AutoML challenge. *2015 International Joint Conference on Neural Networks (IJCNN)*, 1–8. https://doi.org/10.1109/IJCNN.2015.7280767

Hutter, F., Kotthoff, L., & Vanschoren, J. (Eds.). (2019). *Automated machine learning - methods, systems, challenges*. Springer. https://doi.org/10.1007/978-3-030-05318-5

Olson, R. S., & Moore, J. H. (2019). TPOT: A tree-based pipeline optimization tool for automating machine learning. In *Automated machine learning* (pp. 151–160). Springer International Publishing. https://doi.org/10.1007/978-3-030-05318-5_8

Vrbančič, G., Brezočnik, L., Mlakar, U., Fister, D., & Fister, I. (2018). NiaPy: Python microframework for building nature-inspired algorithms. *Journal of Open Source Software*, *3*(23), 613. https://doi.org/10.21105/joss.00613

Yang, X.-S. (2014). Nature-inspired optimization algorithms. In *Nature-inspired optimization algorithms*. Elsevier. https://doi.org/10.1016/b978-0-12-416743-8.00017-8