University of Reading

Enabling automated machine learning for model-driven AI engineering

Article

Accepted Version

Moin, A., Wattanavaekin, U., Lungu, A., Badii, A., Gunnemann, S. and Challenger, M. (2022) Enabling automated machine learning for model-driven AI engineering. IEEE Software. ISSN 1937-4194 (In Press) Available at https://centaur.reading.ac.uk/108613/

It is advisable to refer to the publisher's version if you intend to cite from the work. See Guidance on citing.

Publisher: Institute of Electrical and Electronics Engineers

All outputs in CentAUR are protected by Intellectual Property Rights law, including copyright law. Copyright and IPR is retained by the creators or other copyright holders. Terms and conditions for use of this material are defined in the End User Agreement.

www.reading.ac.uk/centaur

CentAUR

Central Archive at the University of Reading

University of Reading

Reading's research outputs online

Enabling Automated Machine Learning for Model-Driven AI Engineering

Armin Moin, School of Computation, Information, and Technology (CIT), Technical University of Munich (TUM), University of Antwerp & Flanders Make, Germany & Belgium

Ukrit Wattanavaekin and Alexandra Lungu, School of CIT, Technical University of Munich (TUM), Germany

Atta Badii, Department of Computer Science, University of Reading, United Kingdom

Stephan Gunnemann School of CIT & Munich Data Science Institute, Technical University of Munich, Germany

Moharram Challenger, Department of Computer Science, University of Antwerp & Flanders Make, Belgium

Abstract—This article presents our work in progress in supporting automated machine learning in the model-driven engineering process of AI-enabled software systems. We argue that the state of practice suffers from two key issues. First, data scientists often follow a trial-and-error process and use certain heuristics to practice machine learning engineering. Therefore, their results are typically far from optimized as we show through an example in this study. Second, software engineers without deep knowledge of machine learning are often required to collaborate with data scientists, integrate and maintain their code, or even take over their tasks due to a general shortage of data scientists worldwide. Hence, there is an urgent need for tools that can support these novice machine learning practitioners. To address the mentioned issues, we deploy the model-driven engineering paradigm and enable automated machine learning in an existing software development methodology and tool that supports this paradigm.

Introduction

two groups of practitioners serve as the target audience of this study: i) Machine Learning (ML) experts, such as data scientists, who propose ad hoc ML solutions to problems, but their ML models might not possess the most optimized architectures and hyperparameters; moreover, their practice is often not concerned with the architecture of the larger system in which the ML component should be deployed; ii) Software developers who are not ML experts but may occasionally have to carry out certain ML tasks or integrate ML components into larger systems. Several recent studies, such as the work of Lewis et al. [1] on the so-called ML Mismatch problem clearly pointed out part of the mentioned issues. The rest is reflected in our motivating example below.

Prior work in the literature (e.g., see [2], [3], and [4]) proposed deploying the Model-Driven Engineering (MDE) paradigm for AI (specifically ML) engineering. Due to abstraction and automation, domain-specific Model-Driven Software Engineering (MDSE) approaches in the industry have enabled a 300-1,000% productivity leap on average compared to general-purpose modelling or manual software coding [5]. Combined with ML engineering, prior work [2] reported a 236% productivity leap by using the domain-specific MDE approach compared to manual software development and ML practices. Although the full source code in Python and Java would be generated out of the high-level specifications (i.e., ML-enabled software models) in their work [2], ML

practitioners had to still choose the suitable ML model architecture and the hyperparameters. However, in this work, we propose enabling Automated ML (AutoML) to automate and optimize the processes of ML model architecture selection and ML model hyperparameter tuning. In the following, we first present a motivating example from the domain of smart energy systems. We then propose the AutoML-enhanced MDE approach and elaborate on our implementation details. Finally, we conclude, point out the limitations, and refer to future work.

Motivating example

Given an aggregate signal of the electrical energy consumption of a household over a period of time, it is possible to disaggregate the signal into a number of individual loads, thus making educated guesses about the type of electrical appliances inside the household (according to their signatures), as well as their individual power consumption with a reasonable level of accuracy. This is called Non-Intrusive (Appliance) Load

Monitoring (NIALM/NILM) or energy disaggregation. The problem is an inverse problem similar to the Single Channel Blind Source Separation (SCBSS) problem in Physics and Signal Processing. Energy disaggregation has various use cases, such as increasing user awareness, thus improving energy saving and sustainability. The problem has been identified since the 1980s and has recently gained interest in the ML community.

We present the energy disaggregation use case and the proposed approaches to this problem in the literature to motivate our study and illustrate the highlighted issues above. Recently, various ML approaches with rather complex ML models have been proposed in the literature on energy disaggregation which can be easily outperformed by simpler ML models (as we show) that require much fewer resources for training and prediction. However, in the absence of AutoML approaches (or due to refraining from employing such techniques), complex and resource-inefficient methods have become adopted. Additionally, different studies used different datasets, setups, and even metrics to validate their work. Hence, they were not essentially comparable with each other.

To avoid anecdotal claims and enable objective benchmarking, we conducted experiments with the implementations of six proposed approaches in the prior work, as well as three other methods proposed by ourselves, on two reference datasets for energy disaggregation, namely the REDD [6] and the UK-DALE [7] datasets, with similar computational resources, and the same metric, that is the Mean Absolute Error (MAE) for energy disaggregation. We believe that this metric is the most important one among the alternatives reported by various prior work since the key underlying question is how much the predicted energy consumption signal deviates from the actual signal. Also, the advantage of the MAE (i.e., the L1 norm or error) compared to the Mean-Squared Error (MSE), that is the L2 norm (or error), is that it is more robust against outliers. Some of the selected related work methods were already implemented in the open-source NILMTK toolkit [8]. For the ones which were not available, we had to re-implement them on our own since the authors refused to share their code. The results are shown in Table 1.

As we can the easy-to-train and fast-performing Decision Tree (DT) method can outperform the state of the art. Moreover, Gated Recurrent Units (GRUs) ranked second. Further, with an appropriate Hyper-parameter set, the simplest architecture for Artificial Neural Networks (ANNs) could outperform the more complex architectures, such as the DNN-HMMs model [9] or the dictionary-based approach [10], which were both more expensive in terms of computational resources and time.

Table 1. The average Mean Absolute Error (MAE) of the proposed approaches to energy disaggregation

Rank	Approach	MAE
1st	Decision Trees (DTs)	14.86
2nd	Gated Recurrent Units (GRUs)	19.55
3rd	LSTMs (Kelly and Knottenbelt, 2015 [11])	25.77
4th	DAEs (Kelly and Knottenbelt, 2015 [11])	41.44
5th	Fully-Connected Neural Networks (FCNNs)	48.96
6th	Dictionary-based (Elhamifar and Systry, 2015 [10])	61.15
7th	DNN-HMMs (Mauch and Yang, 2016 [9])	118.52
8th	FHMMs (Reyes-Gomez et al., 2003 [12])	139.40
9th	Combinatorial Optimization (Hart, 1992 [13])	204.83

Please note that we excluded some other related work in the literature from our experiments since we believed that they were not practical for real-world use case scenarios. In fact, typical household electricity meters cannot offer high-frequency sampling data (i.e., more than once in a second or 1 Hz). However, many prior works had proposed approaches that were suitable for high-frequency samples which are easier to deal with but more expensive to acquire. In contrast, for our experiments, we adopted the sampling rate of once every 20 seconds (i.e., 0.05 Hz).

Proposed approach

We build the proposed approach on the prior work, called ML-Quadrat[1] [2], which provided a methodology and an open-source modelling tool for creating ML-enabled Internet of Things (IoT) services. In ML-Quadrat, practitioners should specify the target system at the design time using the provided Domain-Specific Language (DSL). Once the model was complete and valid, the full source code of the desired software solution could be generated in a number of programming languages and for several libraries, such as Scikit-Learn and Keras (with the TensorFlow backend). The DSL provided a higher level of abstraction at which the structure and behaviour of the tar-get system including its ML components could be specified in a platform-independent manner. However, the practitioner had to explicitly specify the ML model architecture (e.g., the Multi-Layer Perceptron ANNs) and the hyperparameters (e.g., the learning algorithm, learning rate, weight initialisation, and data preparation techniques). By contrast, in this work, we advocate automating the ML model architecture selection and hyper-parameter tuning processes using AutoML. Figure 1 illustrates part of a sample model instance in ML-Quadrat. The red arrow emphasizes the part which was manually specified by the practitioner, but it should be possible to have this automatically specified (on demand) in the future as a result of the present work.

```
data_analytics da1 {
    labels ON
    features aggregated_load, disaggregated_loads_1
    prediction_results disaggregated_loads_1
    dataset "data/nialm.csv"
    sequential TRUE
    timestamps ON
    preprocess_feature_scaler StandardScaler
    model_algorithm nn_multilayer_perceptron my_nn_mlp(
        activation relu,
        optimizer adam,
        learning_rate_init "0.00001"
    )
    training_results "data/training_da1.txt"
}
```

Figure 1. Part of a sample model instance in ML-Quadrat

Current data science practices involve lots of heuristics and trial-and-error tasks. AutoML aims to automate the practice of ML by offering systematic solutions to non-experts (and experts) in the field of ML in a more efficient manner, compared to the manual approach. The automation might be applied to any part of the ML pipeline: from data pre-processing to hyper-parameter optimization and model evaluation. Most existing AutoML solutions concentrate on one particular part of the ML pipeline, such as Neural Architecture Search (NAS) [14].

The proposed approach in this work is based on two pillars. First, we define and realize various rules according to the guidelines of the API documentation of the target libraries, namely Scikit-Learn and Keras, as well as the ML domain knowledge and best practices in the model-to-code transformation (i.e., code generator) of ML-Quadrat. For instance, the practitioner receives warning or error messages if certain hyper-parameters are out of the recommended or allowed ranges. Moreover, if the AutoML mode is set to on, the rules will be enforced, thus preventing the practitioner from making certain problematic choices, such as shuffling data when the order should be preserved (e.g., deploying the cross-validation technique for sequential data), or re-fusing to standardize numeric data when certain ML models that are sensitive to scale imbalance (e.g., ANNs) should be used.

Second, we offer a standalone, open-source tool, called AutoNIALM[2]. Unlike the first pillar, this one is specific to the problem domain of the example above, namely energy disaggregation. However, it serves as a proof-of-concept that can be extended to cover other use cases and vertical domains too. This tool deploys Bayesian Optimisation (BO) through the Hyperopt library [15]. In fact, AutoNIALM is a domain-specific and low-code workbench that enables practitioners and domain experts in the field of energy disaggregation to select the best ML model architecture and hyperparameters for their specific dataset in an automated manner. This is in line with the so-called citizen data scientist and end-user programmer trends. Once they have the results, they may utilize them in their software model instance in ML-Quadrat. In other words, they can explicitly specify the recommended ML model architecture and hyperparameters using the DSL of ML-Quadrat in the software model instance. Alternatively, they may use the so-called black box ML mode of ML-Quadrat, thus using a pre-trained ML model without specifying the ML component in the software model. This way, they can simply plug the ML model that is trained by the AutoNIALM workbench into the software model instance in ML-Quadrat.

---

[1] https://github.com/arminmoin/ML-Quadrat

Although the Hyperopt Python library is open-source and publicly available, many practitioners do not feel confident to use it. As a result, we see the status quo that we elaborated on through the example above (see Table 1). Therefore, we believe that our workbench may support practitioners in the future in choosing the appropriate ML methods for their energy disaggregation use cases.

The alternative ML methods that can be automatically selected and the possible hyper-parameters to be automatically tuned for each of them are presented in Figure 2. The Hyperopt library requires the depicted tree-structured search space as input to carry out the BO. As shown, we currently support the following methods for energy disaggregation:

1) Decision Trees (DTs),
2) Random Forests (RFs),
3) Gated Recurrent Units (GRUs),
4) Long Short-Term Memories (LSTMs) [11],
5) Fully-Connected Neural Networks (FC-NNs),
6) Denoising Autoencoders (DAEs) [11],
7) Factorial Hidden Markov Models (FH-MMs) [12], and
8) Combinatorial Optimization (CO) [13].

Further, we allow the hyperparameters that are demonstrated in Figure 2 to be tuned automatically as follows:

- criterion ∈ {MSE[3], Friedman MSE, MAE[4]},
- min sample split ∈ Uniform [2,200],
- n estimators[5] ∈ Uniform [5,100],
- optimizer ∈ {Adam, Nadam, RMSprop},
- learning rate ∈ {1e-2, 1e-3, 1e-4, 1e-5},
- loss function ∈ {MSE, MAE},
- n layers[6] ∈ Uniform [5, 8],
- dropout probability ∈ Uniform [0.1, 0.6],
- sequence length ∈ {64, 128, 256, 512, 1024}.

In addition, we enabled three modes in the AutoNIALM workbench: auto, quick, and manual. The auto mode launches a total of 200 trials to find the best ML model architecture and hyperparameters. Also, the number of epochs for training the ML model is set to 2, 000, and the so-called patience rate is set to 15. In contrast, in the quick mode, the total number of trials, the number of training epochs, and the patience rate are set to 50, 500, and 5, respectively. However, in the manual mode, all of the said parameters can be set by the practitioner to the desired values. Finally, regardless of the mode, the data are split into the typical 80% and 20% parts for the training and the validation datasets, respectively. Two screenshots of the AutoNIALM workbench are provided in Figures 3 and 4.

---

2 https://github.com/ukritw/autonialm

[3]MSE stands for the Mean Squared Error.
[4]MAE stands for the Mean Absolute Error.
[5] i.e., number of estimators
[6]i.e., number of layers

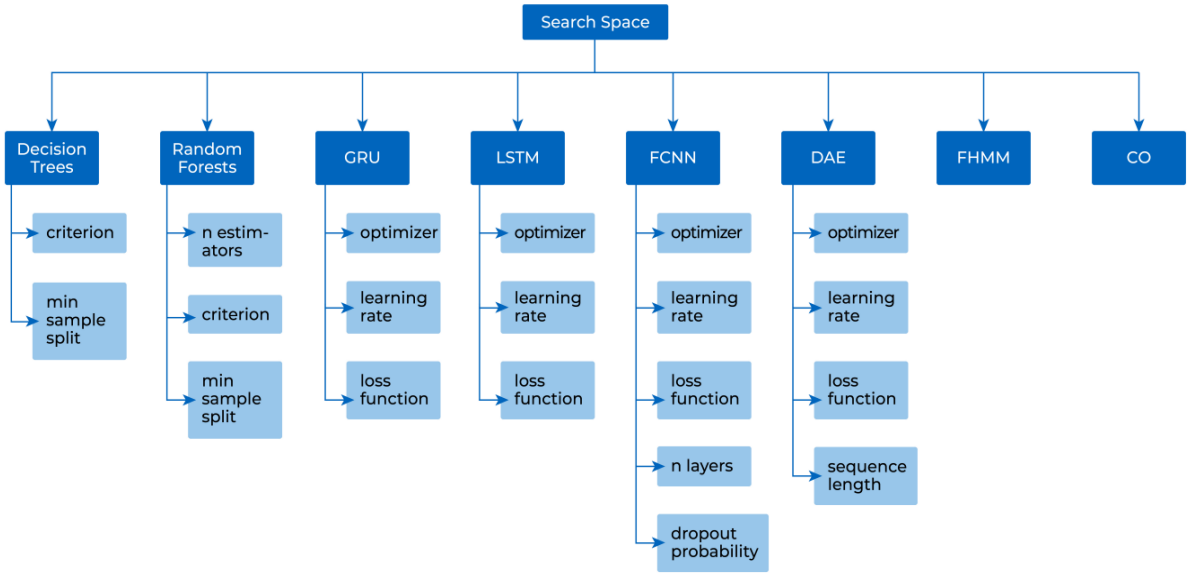


Figure 2. The tree-structured search space for the proposed AutoML approach

Conclusion and future work

In this article, we have presented our work in progress concerning supporting AutoML in model-driven AI (specifically ML) engineering. First, we have shown the need for such an approach through a motivating example. Second, we have addressed this by proposing AutoML in MDE practices for ML engineering, in particular for the use case domain of energy disaggregation. Our preliminary validation has been conducted by using our open-source AutoNIALM workbench for the case of the motivating example (i.e., energy disaggregation using the mentioned reference datasets). It transpired that our tool could find the most suitable ML model architecture and the optimal hyperparameters to outperform the state of the art. However, a more thorough validation study will be required to show the feasibility and efficiency for other datasets, use case scenarios, vertical domains, and even other ML model architectures, methods, and techniques, which were not supported by this study. In particular, an empirical user study, ideally in the form of a randomized controlled experiment, with external practitioners will be required to show the productivity leap that is expected to be the result of deploying the proposed methodology and tooling for developing ML-enabled software systems.

Furthermore, the AutoNIALM workbench and the ML-Quadrat modelling tool are currently loosely coupled. In other words, practitioners need to first use the AutoNIALM workbench and then utilize the results (e.g., the recommended ML model architecture and hyperparameters, or the trained ML model) in ML-Quadrat. Future work will integrate these projects so that ML-Quadrat can serve as a one-stop-shop for practitioners.

References

1. Lewis, G.A., Bellomo, S., Ozkaya, I. (2021). Characterizing and Detecting Mismatch in Machine-Learning-Enabled Systems. IEEE/ACM 1st Workshop on AI Engineering - Software Engineering for AI (WAIN), 133-140. https://doi.org/10.1109/WAIN52551.2021.00028.

2. Moin, A., Challenger, M., Badii, A., Guineaman, S. (2022). A model-driven approach to machine learning and software modelling for the IoT. Software and Systems Modelling (SoSyM). Vol. 21, No. 3. pp. 987-1014. https://doi.org/10.1007/s10270-021-00967-x.

3. Hartmann, T., Moawad, A., Fouquet, F. et al. (2019). The next evolution of MDE: a seamless integration of machine learning into domain modelling. Software and Systems Modelling (SoSyM) Vol. 18, No. 2, pp. 1285–1304. https: //doi.org/10.1007/s10270-017-0600-2.

4. Bishop, C. M. (2013). Model-Based Machine Learning. Philosophical Transactions of the Royal Society A. Vol. 371, No. 1984. https://doi.org/10.1098/rsta.2012.0222.

5. Kelly, S., Tolvanen, J.P. (2008). Domain-Specific Modelling: Enabling Full Code Generation, 1st edn. Wiley, Hoboken.

6. Kolter, J. Z., Johnson, M. J. (2011). REDD: A public data set for energy disaggregation research. Proc. of the SustKDD workshop on Data Mining Applications in Sustainability.

7. Kelly, J., Knottenbelt, W. (2015). The UK-DALE dataset, domestic appliance-level electricity demand and whole-house demand from five UK homes. Sci Data 2, 150007. Nature.

8. Batra, N., Kelly, J., Parson, O. et al. (2014). NILMTK: an open-source toolkit for non-intrusive load monitoring. In Proceedings of the 5th international conference on Future energy systems (e-Energy '14). ACM, New York, NY, USA, pp. 265–276. https://doi.org/10.1145/2602044. 2602051.

9. Mauch, L., Yang, B. (2016). A novel DNN-HMM-based approach for extracting single loads from aggregate power signals. Proc. of the IEEE ICASSP.

10. Elhamifar, E., Sastry, S. (2015). Energy disaggregation via learning 'Powerlets' and sparse coding. Proc. of the AAAI Conf. on Artificial Intelligence.

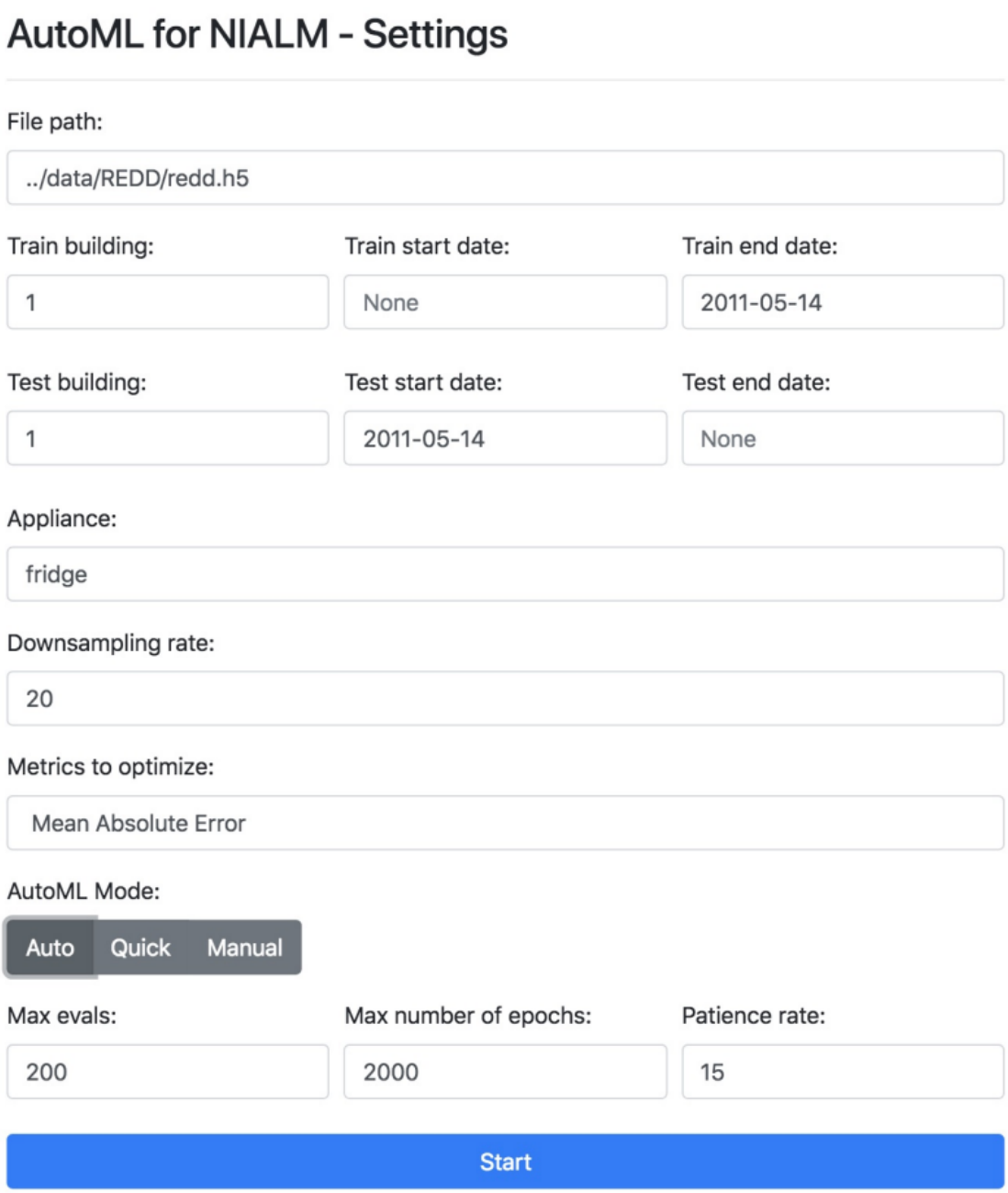
11. Kelly, J., Knottenbelt, W. (2015). Neural NILM: Deep Neural Networks Applied to Energy Disaggregation. Proc. of the ACM Int. Conf. on Embedded Systems for Energy-Efficient Built Environments.

12. Reyes-Gomez, M. J., Raj, B., Ellis, D. R. W. (2003). Multi-channel source separation by factorial HMMs. Proc. of the IEEE ICASSP.

13. Hart, G. W. (1992). Nonintrusive appliance load monitoring. Proc. of the IEEE, vol. 80, no. 12, pp. 1870-1891.

14. He, X., Zhao, K., Chu, X. (2021). "AutoML: A survey of the state-of-the-art" Knowledge-Based Systems., Volume 212.

15. Bergstra, J., Yamins, D., Cox, D. D. (2013) Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. Proc. of ICML.



16.

Figure 3. A screenshot of the AutoNIALM workbench: the input form for practitioners

AutoML Result

Trial Options and Metadata

hd5_filepath: data/REDD/redd.h5
train_building: 1
train_start: None
train_end: 2011-05-14
test_building: 1
test_start: 2011-05-14
test_end: None
appliance: fridge
downsampling_period: 20
num_epochs: 2000
patience: 15
max_evals: 200
metrics_to_optimize: mean_absolute_error
trial_start_date: 2018-09-03
trial_start_time: 00:49:27.328060
time_taken: 73129.60
best algorithm: {'criterion': 'mae', 'min_samples_split': 196.0, 'n_estimators': 24.0, 'type': 'random forest'}
best loss: 15.87

Leaderboard

Rank	Algorithm	loss(mean_absolute_error)	Model Information
1	**random forest**	**15.87**	Info
	Hyperparameters: criterion: mae min_samples_split: 196.0 n_estimators: 24.0 type: random forest	Evaluation metrics accuracy_score: 0.94 f1_score: 0.91 precision_score: 0.87 recall_score: 0.96 mean_absolute_error: 15.87 mean_squared_error: 2494.92 relative_error_in_total_energy: 0.07 nad: 0.48 disaggregation_accuracy: 0.89	Time taken: 533.82s
2	**random forest**	**15.87**	Info
3	**random forest**	**15.91**	Info
4	**random forest**	**15.92**	Info

Figure 4. A screenshot of the AutoNIALM workbench: the output results of AutoML