Time Series Machine Learning with aeon: Classification and Regression

Matthew Middlehurst[1,2] ✉, Anthony Bagnall[1,3], Germain Forestier[4,5], Ali Ismail-Fawaz[4], and Antoine Guillaume[6]

[1] University of Southampton, Southampton, United Kingdom
[2] University of Bradford, Bradford, United Kingdom
[3] University of East Anglia, Norwich, United Kingdom
[4] IRIMAS, Université de Haute-Alsace, Mulhouse, France
[5] DSAI, Monash University, Melbourne, Australia
[6] Novahé, France

Abstract. We present the classification and regression modules of `aeon`, a Python library for all machine learning tasks involving time series. `aeon` follows the `scikit-learn` API and is compatible with its utilities such as model selection and pipelines. The toolkit contains a wide range of algorithms, including the state-of-the-art and popular benchmarks for each time series learning task. We demonstrate how to use the `aeon` toolkit for these tasks and give an example of where these algorithms may be useful. More information and an introductory video of the toolkit modules are available on the demo webpage `https://aeon-tutorials.github.io/ECML-Demo-2025/`.

Keywords: Time series · Classification · Regression · aeon

1 Introduction

Time series machine learning (TSML) is an active research field that finds areas of application in all domains of machine learning and data science. The largest and most well-known field of research for time series is forecasting, having an abundance of tutorials and software suites available to prospective users such as `statsmodels`, `statsforecast`, `darts`, `kats`, and `greykite`. This has left the more traditional machine learning tasks such as classification and regression with a comparative lack of resources despite also being a thriving area of research [12,4,13,3].

Time series classification (TSC) [12] involves fitting a model from a continuous, ordered sequence of real valued observations (a time series) to a discrete response variable. Time series extrinsic regression (TSER) [4] uses a continuous response variable. TSC and TSER problems come from various domains, including medical signals such as electrocardiography (ECG) and Electroencephalography (EEG); human activity recognition and other motion data; spectrograms; audio; electricity usage; and many more with industrial and research applications. For simplicity, we refer to the grouping of these learning tasks as TSML.

In this demonstration, we show simple use case examples of algorithms for each task and demonstrate how `aeon` is interoperable with `scikit-learn`. We describe modules from the `aeon` toolkit that fill the current resource gap and provide easy access to the most recent TSML algorithms and tools.

2 The `aeon` Toolkit

The `aeon` [9] toolkit is an open-source framework for all time series learning tasks. `aeon` estimators extend the `scikit-learn BaseEstimator` framework, retaining the familiar `fit` and `predict` structure. The `aeon BaseClassifier` and `BaseRegressor` base classes inherited by TSML algorithms are used to process data and provide common utilities. `aeon` provides efficient implementations of the latest advances for a broad range of TSML tasks, including modules for classification, regression, clustering, forecasting, anomaly detection, segmentation and similarity search, as well as a variety of utilities, transformations, and distance measures designed for time series data. In this demonstration we will provide an overview of the functionality available for classification and regression. The toolkit supports both univariate and multivariate (if the algorithm allows) time series, with growing support and utilities for unequal length data. Using a system of optional dependencies, `aeon` integrates a wide variety of packages such as the previously mentioned TSML projects into a single interface while keeping the core framework with minimal dependencies. Our aim is to provide optimised implementations wherever possible using `numba`.

The toolkit follows the taxonomy presented in a recent comparative study [12] for its design of the classification and regression modules. Algorithms are grouped into packages of the following categories: convolutions, deep learning, dictionaries/bags-of-words, distance functions, feature extraction, random intervals, shapelets and hybrids. Two recent studies showed that HIVE-COTEv2 [11] and MultiROCKET-HYDRA [2,15] were the state-of-the-art for TSC using the UCR archive datasets [1]; and FreshPRINCE [7], DrCIF [11,10] and RIST [8] were the state-of-the-art for TSER on the Monash/UEA regression archive [4,8,14]. `aeon` contains implementations of all these algorithms, as well as a majority of the approaches they were compared against.

The package is open-source and distributed under the 3-Clause BSD license. The source code is shared on GitHub[1]. New additions to the codebase are thoroughly tested, with code quality standards upheld. The package is distributed through both `pip`[2] and `conda-forge`. The project documentation is `sphinx` based, hosted by Read the Docs and is available at `https://aeon-toolkit.org`.

2.1 Other TSML packages

Other packages which can be used for TSML in the Python ecosystem include `tslearn` and `pyts`. However, these present a smaller range of simpler algorithms

[1] `https://github.com/aeon-toolkit/aeon`
[2] `https://pypi.org/project/aeon/`

and lack the state-of-the-art and latest advances. Both packages contain under ten algorithms at the time of writing, compared to the 80+ implementations available in the `aeon` classification and regression modules. The `tsai` package includes deep learning models usable for TSML, but is missing more recent approaches [6,5] available in `aeon` and does not use the `scikit-learn` API.

Other packages such as `tsfresh` and `stumpy` focus on a single algorithm or set of features to extract. While effective, they are not toolkits. The framework in such packages has been developed to best fit the algorithm rather than a range of approaches. This can cause friction when including them in evaluations of multiple approaches, introducing issues such as differing input layouts and results formats which can be avoided with the `aeon` framework.

3 Using `aeon`

We provide some simple usage examples for selected algorithms for each of the `aeon` TSML modules using datasets from the UCR [1] and Monash/UEA [4,14] archives. A list of available estimators is available on the `aeon` webpage. Further examples pertaining to the modules discussed such as data formatting and loading are available on the repository and the demo webpage. The following code is from version 1.2.0 of `aeon`.

3.1 Classification

The classification module workflow will be familiar to anyone who has worked with `scikit-learn` previously. We use the `load_classification` function to download the ArrowHead UCR archive dataset. We train a `MultiRocketHydraClassifier` on the loaded data using the `fit` method, which requires the training data `X` and the training labels `y`. Following this, the `predict_proba` method is used to predict the class probabilities for each case in `X_test`. To directly predict class labels, the `predict` method can be used.

```
from aeon.datasets import load_classification
from aeon.classification.convolution_based import
    MultiRocketHydraClassifier
X, y = load_classification("ArrowHead", split="TRAIN")
X_test, _ = load_classification("ArrowHead", split="TEST")
clf = MultiRocketHydraClassifier().fit(X, y)
probabilities = clf.predict_proba(X_test)
```

3.2 Regression

The regression module is similar to classification in its workflow, but numeric labels are required and finding label probabilities with `predict_proba` is not supported. We load the BarCrawl6min dataset from the Monash archive and fit a `DrCIFRegressor` on it. Label predictions are found using the `predict` method.

```
from aeon.datasets import load_regression
from aeon.regression.interval_based import DrCIFRegressor
X, y = load_regression("BarCrawl6min", split="TRAIN")
X_test, _ = load_regression("BarCrawl6min", split="TEST")
reg = DrCIFRegressor(n_estimators=100).fit(X, y)
predictions = reg.predict(X_test)
```

4 Contributing to `aeon`

The `aeon` package welcomes code contributions via Pull Requests on GitHub. The community strives to maintain a consistent style and framework throughout the codebase, but can be flexible in accommodating other approaches. The primary requirement for contributing a new algorithm is that it must have been published in a reputable venue. Comprehensive contribution and developer guidelines are available in the documentation[3], and developers are available to guide researchers who want to submit their published algorithms. The `aeon` framework can be easily extended by inheriting the `BaseClassifier` or `BaseRegressor` abstract classes and implementing the abstract methods within. The core framework manages tasks such as data validation and conversion, metadata tag handling, as well as integration with the `scikit-learn` API. The codebase contains extensive documentation and automatic testing to avoid common implementation errors.

5 Conclusions

We present the classification and regression modules of the `aeon` toolkit. `aeon` innovates through its breadth of coverage, its ease of use and inclusion of efficient implementations of the latest state-of-the-art algorithms. Our primary goal with `aeon` is to reduce the lead time between the publication of a description of an algorithm and its availability to practitioners to use for solving real-world problems. We believe it is a useful tool for researchers wanting to explore new algorithms. We believe `aeon` has three sets of target users. Firstly, researchers implementing their algorithms within the framework can help extend the impact of their work through reproducible research and easy comparison to the current state of the art. Secondly, scientists and those in industry benefit from a reduced lead time between the proposing of a new technique and its easy integration into existing workflows. Finally, our easy-to-use and familiar `scikit-learn` interface makes `aeon` a useful teaching tool for TSML and reduces barriers to entry into the research field. The `aeon` package is open to all, integrated into the Python ecosystem, and a big step toward reproducible research for TSML.

Acknowledgments. This work is supported by the UK Engineering and Physical Sciences Research Council (EPSRC) grant number EP/W030756/2.

[3] `https://www.aeon-toolkit.org/en/stable/developer_guide.html`

References

1. Dau, H.A., Bagnall, A., Kamgar, K., Yeh, C.C.M., Zhu, Y., Gharghabi, S., Ratanamahatana, C.A., Keogh, E.: The ucr time series archive. IEEE/CAA Journal of Automatica Sinica **6**(6), 1293–1305 (2019)
2. Dempster, A., Schmidt, D.F., Webb, G.I.: Hydra: Competing convolutional kernels for fast and accurate time series classification. Data Mining and Knowledge Discovery **37**(5), 1779–1805 (2023)
3. Fawaz, H.I., Forestier, G., Weber, J., Idoumghar, L., Muller, P.A.: Deep learning for time series classification: a review. Data Mining and Knowledge Discovery **33**(4), 917–963 (2019)
4. Guijo-Rubio, D., Middlehurst, M., Arcencio, G., Silva, D.F., Bagnall, A.: Unsupervised feature based algorithms for time series extrinsic regression. Data Mining and Knowledge Discovery (2024)
5. Ismail-Fawaz, A., Devanne, M., Berretti, S., Weber, J., Forestier, G.: Look into the lite in deep learning for time series classification. International Journal of Data Science and Analytics pp. 1–21 (2025)
6. Ismail-Fawaz, A., Devanne, M., Weber, J., Forestier, G.: Deep learning for time series classification using new hand-crafted convolution filters. In: 2022 IEEE International Conference on Big Data (Big Data). pp. 972–981. IEEE (2022)
7. Middlehurst, M., Bagnall, A.: The freshprince: A simple transformation based pipeline time series classifier. In: International Conference on Pattern Recognition and Artificial Intelligence. pp. 150–161. Springer (2022)
8. Middlehurst, M., Bagnall, A.: Extracting features from random subseries: A hybrid pipeline for time series classification and extrinsic regression. In: International Workshop on Advanced Analytics and Learning on Temporal Data. pp. 113–126. Springer (2023)
9. Middlehurst, M., Ismail-Fawaz, A., Guillaume, A., Holder, C., Guijo-Rubio, D., Bulatova, G., Tsaprounis, L., Mentel, L., Walter, M., Schäfer, P., et al.: aeon: a python toolkit for learning from time series. Journal of Machine Learning Research **25**(289), 1–10 (2024)
10. Middlehurst, M., Large, J., Bagnall, A.: The canonical interval forest (cif) classifier for time series classification. In: 2020 IEEE international conference on big data (big data). pp. 188–195. IEEE (2020)
11. Middlehurst, M., Large, J., Flynn, M., Lines, J., Bostrom, A., Bagnall, A.: Hivecote 2.0: a new meta ensemble for time series classification. Machine Learning **110**(11), 3211–3243 (2021)
12. Middlehurst, M., Schäfer, P., Bagnall, A.: Bake off redux: a review and experimental evaluation of recent time series classification algorithms. Data Mining and Knowledge Discovery (2024)
13. Ruiz, A.P., Flynn, M., Large, J., Middlehurst, M., Bagnall, A.: The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances. Data Mining and Knowledge Discovery **35**(2), 401–449 (2021)
14. Tan, C.W., Bergmeir, C., Petitjean, F., Webb, G.I.: Time series extrinsic regression: Predicting numeric values from time series data. Data Mining and Knowledge Discovery **35**, 1032–1060 (2021)
15. Tan, C.W., Dempster, A., Bergmeir, C., Webb, G.I.: Multirocket: Multiple pooling operators and transformations for fast and effective time series classification. Data Mining and Knowledge Discovery pp. 1–24 (2022)