Journal of Machine Learning Research 24 (2023) 1-6 Submitted 7/22; Revised 3/23; Published 6/23

Merlion: End-to-End Machine Learning for Time Series

Aadyot Bhatnagar*, Paul Kassianik*, Chenghao Liu*, Tian Lan*, Wenzhuo Yang*, Rowan Cassius†, Doyen Sahoo*, Devansh Arpit*, Sri Subramanian†, Gerald Woo*, Amrita Saha*, Arun Kumar Jagota†, Gokulakrishnan Gopalakrishnan‡, Manpreet Singh‡, K C Krithika‡, Sukumar Maddineni†, Daeki Cho§, Bo Zong§, Yingbo Zhou*, Caiming Xiong*, Silvio Savarese*, Steven Hoi*, Huan Wang*

Editor: Antti Honkela

Abstract

We introduce Merlion, an open-source machine learning library for time series. It features a unified interface for many commonly used models and datasets for forecasting and anomaly detection on both univariate and multivariate time series, along with standard pre/post-processing layers. It has several modules to improve ease-of-use, including a no-code visual dashboard, anomaly score calibration to improve interpetability, AutoML for hyperparameter tuning and model selection, and model ensembling. Merlion also provides an evaluation framework that simulates the live deployment of a model in production, and a distributed computing backend to run time series models at industrial scale. This library aims to provide engineers and researchers a one-stop solution to rapidly develop models for their specific time series needs and benchmark them across multiple datasets.

Keywords: time series, forecasting, anomaly detection, machine learning, autoML, ensemble learning, benchmarking, Python, scientific toolkit

1. Introduction

Time series are ubiquitous in monitoring the behavior of complex systems in real-world applications, such as IT operations management, manufacturing, and cybersecurity (Hundman et al., 2018; Mathur and Tippenhauer, 2016; Audibert et al., 2020). Across all these applications, it is important to accurately forecast the trends and values of key metrics and to rapidly and accurately detect anomalies in those metrics.

This work introduces Merlion[1], a Python library for time series intelligence. It provides an end-to-end machine learning framework that includes loading and transforming data, building and training models, post-processing model outputs, evaluating model performance, scalable deployment using a distributed back-end, and a clickable visual dashboard for code-free experimentation. It supports unified APIs for a diverse set of machine learning models for forecasting, anomaly detection, and change point detection on both univariate and multivariate time series. Supported models range from classical models like ARIMA to Bayesian methods to deep learning models.

*. AI Research, Salesforce. Corresponding Authors: {shoi,huan.wang}@salesforce.com
†. Monitoring Cloud, Salesforce
‡. Warden AIOps, Salesforce
§. Service Protection, Salesforce
1. Code: https://github.com/salesforce/Merlion. Long paper: https://arxiv.org/abs/2109.09265.

Several other time series libraries support similar models (Seabold and Perktold, 2010; Taylor and Letham, 2017; Van Looveren et al., 2019; Alexandrov et al., 2020; Jiang, 2021; Hosseini et al., 2021; Herzen et al., 2022). However, Merlion offers the most complete end-to-end solution. In particular, Merlion has the best support for ensembling heterogeneous models, post-processing model outputs, visualizing model performance, and deploying scalable applications using distributed computing.[2]

2. Design Principles and Key Components

Merlion's key design principles are modularity, extensibility[3], and ease of use. Merlion is divided into five modules that can easily be composed with each other: data, models, post-processing, ensembles & model selection, and evaluation. The codebase is heavily object-oriented, and it leverages powerful base classes to minimize the amount of code needed to extend the library with custom datasets, models, pre/post-processing, or evaluation metrics. These base classes also provide many convenience features including visualization and serialization. Finally, we provide a clickable visual dashboard that enables code-free experimentation, as well as a distributed back-end that uses pySpark (Zaharia et al., 2016) and Kubernetes to deploy time series application at industrial scale.

Data Merlion's `TimeSeries` data structure represents a d-dimensional time series T as a collection of univariates $U^{(1)}, \dots, U^{(d)}$, where $U^{(k)} = (t_1^{(k)}, x_1^{(k)}), \dots, (t_{n_k}^{(k)}, x_{n_k}^{(k)})$. This reflects the reality that individual univariates may be sampled at different rates, or contain missing data at different timestamps. Users can create `TimeSeries` objects directly from files on disk, `pandas` dataframes, or Spark distributed datasets. We also provide standardized loaders for a wide range of datasets in the `ts_datasets` package.

After the data is loaded, the `merlion.transform` module lets users pre-process their data in various ways that can make it easier to model. These include resampling, normalization, moving averages, temporal differencing, and many others. Users may also extend this module with other data processing tools (Christ et al., 2018; Law, 2019). Multiple transforms can be composed with each other (e.g. resampling followed by a moving average), and transforms can be inverted (e.g. the normalization $f(x) = (x - \mu)/\sigma$ is inverted as $f^{-1}(y) = \sigma y + \mu$). All models have a `model.transform` which is automatically applied to any input data.

Models Merlion provides unified APIs for a wide array of forecasting and anomaly detection models. These include statistical methods, tree-based models, deep learning approaches, and others. We also provide easy-to-use autoML for automatic hyperparameter selection.

All models are initialized with a `config` object that contains implementation-specific hyperparameters, and support a `model.train(time_series)` method. Given a general multivariate time series $T = (U^{(1)}, \dots, U^{(d)})$, forecasters predict the values of a single target univariate $U^{(k)}$. One can obtain a model's forecast of $U^{(k)}$ by calling `model.forecast(time_stamps)`. Analogously, one can obtain an anomaly detector's sequence of anomaly scores by calling `model.get_anomaly_score(time_series)`. Forecasters can be used for anomaly detection by treating the residual between the true and predicted value of a target $U^{(k)}$ as an anomaly score. We implement change point detection as a specific instance of anomaly detection.

2. See Merlion's README file for a more detailed comparison.
3. See Merlion's contributing guidelines for detailed information about extending Merlion.

All models can also condition their predictions on historical data without having to be re-trained from scratch, by calling `model.forecast(time_stamps, time_series_prev)` or `model.get_anomaly_score(time_series, time_series_prev)`. Finally, many forecasters support *exogenous regressors*, variables whose future values are known a priori, by calling `model.forecast(time_stamps, exog_data=exog_data)`.

Post-Processing All anomaly detectors have a `post_rule` which applies *calibration* and *thresholding* to their outputs. Calibration, a feature unique to Merlion, uses optimal transport to ensure that the anomaly scores follow a standard normal distribution, ensuring that they are both interpretable and consistent across different models. Intelligent thresholding rules reduce a model's false positive rate suppressing redundant alerts. A user can obtain post-processed anomaly scores from a model by calling `model.get_anomaly_label(time_series)`. We refer an interested reader to Bhatnagar et al. (2021, Section 4.2) for more details.

Ensembles and Model Selection Ensembles are structured as a model that represents a combination of multiple underlying models. For this purpose we have a base `EnsembleBase` class that abstracts the process of obtaining predictions $Y_1, \ldots, Y_m$ from m underlying models on a single time series T, and a `Combiner` class that then combines results $Y_1, \ldots, Y_m$ into the output of the ensemble. These combinations include traditional mean ensembles, as well as model selection based on evaluation metrics like sMAPE. Subclasses implement the `forecast` or `get_anomaly_score` methods, and their `train` method automatically handles dividing the data into training and validation splits if performing model selection.

Since most anomaly detectors return vastly different scores, Merlion uses calibration to transform these scores into standard normal random variables. These calibrated scores can then be combined with a simple mean. Merlion is thus the only library to support ensembles of heterogeneous anomaly detectors. See Bhatnagar et al. (2021, Section 4.3) for more details.

Evaluation When a time series model is deployed live in production, training and inference are rarely performed in batch on a full time series. Rather, the model is re-trained at a regular cadence, and where possible, inference is performed in a streaming mode. Merlion's evaluation framework simulates this workflow to obtain predictions from a model that more closely mirror real-world scenarios. We also implement a wide range of evaluation metrics for both forecasting or anomaly detection which conform to a unified interface, and we provide scripts which allow users to easily evaluate model performance on any dataset included in the `ts_datasets` module, as well as custom datasets provided by the user.

Additional Interfaces To ease initial experimentation, Merlion provides a unique no-code visual dashboard which allows users to try different models on custom datasets, and view both quantitative and qualitative evaluations in a single window. Merlion also provides a distributed computing backend using PySpark, which makes it easy to deploy any Merlion model at industrial scale. Only one other library has a distributed backend, and its scope is limited to just a few forecasting models (Garza, 2021).

3. Experiments

In this section, we use Merlion to benchmark the performance of various models on both forecasting (Table 1) and anomaly detection (Table 2). The purpose of this section is not

	m4-hour	m4-day	m4-week	m4-month	m4-quarter	m4-year	internal1	internal2	internal3
MSES	32.45	5.87	16.53	25.40	19.03	21.63	33.50	32.30	3.882
ARIMA	33.54	3.23	9.29	17.66	13.37	16.37	24.00	25.89	5.94
Prophet	18.08	11.67	19.98	20.64	24.53	30.23	56.07	30.93	72.55
ETS	42.95	**3.04**	9.00	14.32	11.08	16.43	17.02	25.10	3.55
AutoSARIMA	**13.61**	3.29	**8.30**	14.26	10.51	17.16	**15.98**	**17.45**	3.42
AutoProphet	16.49	11.67	20.01	20.43	24.62	30.23	56.43	26.35	69.32
AutoETS	19.23	3.07	9.32	**13.73**	**10.33**	**15.96**	21.65	19.41	**3.15**

Table 1: Mean sMAPE achieved by univariate forecasting models on M4 (Makridakis et al., 2018) and 3 internal datasets. We compare ARIMA, Prophet (Taylor and Letham, 2017), ETS, and MSES (Cassius et al., 2021), and our own AutoML variants of ARIMA, Prophet, and ETS. Best results are in **bold**.

	internal	NAB Lavin and Ahmad (2015)	AIOps aio (2018)	UCR Dau et al. (2018)	Δ F1 (vs. best)
ARIMA	**0.531**	0.395	0.227	0.313	0.148 ± 0.099
AutoETS	0.296	0.350	0.097	0.334	0.245 ± 0.042
AutoProphet	0.343	0.323	0.310	0.418	0.166 ± 0.062
Isolation Forest (Liu et al., 2008)	0.436	0.244	0.347	0.461	0.142 ± 0.111
RRCF (Guha et al., 2016)	0.248	0.337	0.314	**0.568**	0.148 ± 0.132
Spectral Residual (Ren et al., 2019)	0.340	0.153	0.338	0.469	0.189 ± 0.150
WindStats (ours, baseline)	0.225	0.247	0.324	0.306	0.239 ± 0.114
ZMS (ours, baseline)	0.486	0.290	0.340	0.427	0.129 ± 0.094
Ensemble (ours)	0.500	**0.548**	**0.396**	0.476	$\mathbf{0.034 \pm 0.044}$

Table 2: F1 scores achieved by univariate anomaly detection models. The ensemble takes the mean *calibrated* anomaly scores of AutoETS, RRCF, and ZMS. We also report the average gap (over datasets) in F1 between each model and the best model. Best results are in **bold**.

to obtain state-of-the-art results. Rather, we highlight the wide range of strong baselines from the literature that Merlion supports. To avoid the possible risk of label leaking through manual hyperparameter tuning, for all experiments, we evaluate all models with a single choice of sensible default hyperparameters and data pre-processing, regardless of dataset.

For each task, we first train an initial model on the training split of a time series, and then re-train the model unsupervised either daily or hourly on the full data until that point (without adjusting the calibrator or threshold). We then *incrementally* obtain predictions for the full time series, in a way that simulates a live deployment scenario. Table 1 shows that Merlion's AutoML module is effective at improving the performance of multiple different forecasting models; Table 2 shows that our proposed ensemble method (a unique offering from Merlion) robustly achieves strong anomaly detection performance. We refer an interested reader to Bhatnagar et al. (2021, Section 5) for additional experiments and more details.

4. Conclusion

We introduce Merlion, an open-source machine learning library for time series, which is designed to address many of the pain points in today's industry workflows for time series anomaly detection and forecasting. It provides unified, easily extensible interfaces and implementations for a wide range of models and datasets, an autoML module that consistently improves the performance of multiple forecasting models, post-processing rules for anomaly detectors that improve interpretability and reduce the false positive rate, and transparent support for ensembles that robustly achieve good performance on multiple benchmark datasets. These features are tied together in a flexible pipeline that quantitatively evaluates the performance of a model, and a visualization module for qualitative analysis.

References

AIOps Challenge, 2018. URL http://iops.ai/competition_detail/?competition_id=5.

A. Alexandrov, K. Benidis, M. Bohlke-Schneider, V. Flunkert, J. Gasthaus, T. Januschowski, D. C. Maddix, S. Rangapuram, D. Salinas, J. Schulz, L. Stella, A. C. Türkmen, and Y. Wang. GluonTS: Probabilistic and Neural Time Series Modeling in Python. *Journal of Machine Learning Research*, 21(116):1–6, 2020. URL http://jmlr.org/papers/v21/19-820.html.

J. Audibert, P. Michiardi, F. Guyard, S. Marti, and M. A. Zuluaga. Usad: Unsupervised anomaly detection on multivariate time series. In *The 26th ACM SIGKDD Intl Conference on Knowledge Discovery & Data Mining*, KDD'20, page 3395–3404, 2020.

A. Bhatnagar, P. Kassianik, C. Liu, T. Lan, W. Yang, R. Cassius, D. Sahoo, D. Arpit, S. Subramanian, G. Woo, A. Saha, A. K. Jagota, G. Gopalakrishnan, M. Singh, K. C. Krithika, S. Maddineni, D. Cho, B. Zong, Y. Zhou, C. Xiong, S. Savarese, S. Hoi, and H. Wang. Merlion: A machine learning library for time series, 2021.

R. Cassius, A. K. Jagota, and A. Bhatnagar. Multi-scale exponential smoother, 2021. URL https://opensource.salesforce.com/Merlion/latest/merlion.models.forecast.html#module-merlion.models.forecast.smoother.

M. Christ, N. Braun, J. Neuffer, and A. W. Kempa-Liehr. Time series feature extraction on basis of scalable hypothesis tests (tsfresh – a python package). *Neurocomputing*, 307: 72–77, 2018. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2018.03.067. URL https://www.sciencedirect.com/science/article/pii/S0925231218304843.

H. A. Dau, E. Keogh, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, Yanping, B. Hu, N. Begum, A. Bagnall, A. Mueen, G. Batista, and Hexagon-ML. The ucr time series classification archive, October 2018. https://www.cs.ucr.edu/~eamonn/time_series_data_2018/.

F. Garza. nixtla. https://github.com/Nixtla/nixtla, 2021.

S. Guha, N. Mishra, G. Roy, and O. Schrijvers. Robust random cut forest based anomaly detection on streams. In M. F. Balcan and K. Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 2712–2721, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/guha16.html.

J. Herzen, F. Lässig, S. G. Piazzetta, T. Neuer, L. Tafti, G. Raille, T. V. Pottelbergh, M. Pasieka, A. Skrodzki, N. Huguenin, M. Dumonal, J. Kościsz, D. Bader, F. Gusset, M. Benheddi, C. Williamson, M. Kosinski, M. Petrik, and G. Grosch. Darts: User-friendly modern machine learning for time series. *Journal of Machine Learning Research*, 23(124): 1–6, 2022. URL http://jmlr.org/papers/v23/21-1177.html.

R. Hosseini, A. Chen, K. Yang, S. Patra, and R. Arora. Greykite: a flexible, intuitive and fast forecasting library, 2021. URL https://github.com/linkedin/greykite.

K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Söderström. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. *CoRR*, abs/1802.04431, 2018. URL `http://arxiv.org/abs/1802.04431`.

X. Jiang. Kats, 2021. URL `https://github.com/facebookresearch/Kats`.

A. Lavin and S. Ahmad. Evaluating real-time anomaly detection algorithms - the numenta anomaly benchmark. *CoRR*, abs/1510.03336, 2015. URL `http://arxiv.org/abs/1510.03336`.

S. M. Law. STUMPY: A Powerful and Scalable Python Library for Time Series Data Mining. *The Journal of Open Source Software*, 4(39):1504, 2019.

F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422, 2008. doi: 10.1109/ICDM.2008.17.

S. Makridakis, E. Spiliotis, and V. Assimakopoulos. The m4 competition: Results, findings, conclusion and way forward. *International Journal of Forecasting*, 34(4):802–808, 2018.

A. P. Mathur and N. O. Tippenhauer. Swat: a water treatment testbed for research and training on ics security. In *2016 International Workshop on Cyber-physical Systems for Smart Water Networks (CySWater)*, pages 31–36, 2016.

H. Ren, B. Xu, Y. Wang, C. Yi, C. Huang, X. Kou, T. Xing, M. Yang, J. Tong, and Q. Zhang. Time-series anomaly detection service at microsoft. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 3009–3017, New York, NY, USA, 2019. Association for Computing Machinery. doi: 10.1145/3292500.3330680.

S. Seabold and J. Perktold. statsmodels: Econometric and statistical modeling with python. In *9th Python in Science Conference*, 2010.

S. J. Taylor and B. Letham. Forecasting at scale. *PeerJ Preprints*, 5(e3190v2), Sept 2017. doi: 10.7287/peerj.preprints.3190v2.

A. Van Looveren, G. Vacanti, J. Klaise, A. Coca, and O. Cobb. Alibi detect: Algorithms for outlier, adversarial and drift detection, 2019. URL `https://github.com/SeldonIO/alibi-detect`.

M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, M. J. Franklin, A. Ghodsi, J. Gonzalez, S. Shenker, and I. Stoica. Apache Spark: A unified engine for big data processing. *Communications of the ACM*, 59(11): 56–65, nov 2016. ISSN 0001-0782. doi: 10.1145/2934664.