Journal of Machine Learning Research 23 (2022) 1-6 Submitted 10/21; Revised 2/22; Published 3/22

Darts: User-Friendly Modern Machine Learning for Time Series

Julien Herzen† julien.herzen@unit8.com
Francesco Lässig† francesco.laessig@unit8.com
Samuele Giuliano Piazzetta samuele.piazzetta@gmail.com
Thomas Neuer† thomas.neuer@unit8.com
Léo Tafti leotafti@gmail.com
Guillaume Raille† guillaume.raille@unit8.com
Tomas Van Pottelbergh† tomas.vanpottelbergh@unit8.com
Marek Pasieka† marek.pasieka@unit8.com
Andrzej Skrodzki andrzej.skrodzki@gmail.com
Nicolas Huguenin† nicolas.huguenin@unit8.com
Maxime Dumonal† maxime.dumonal@unit8.com
Jan Kościsz koscis.j@gmail.com
Dennis Bader† dennis.bader@unit8.com
Frédérick Gusset gusset.frederick@gmail.com
Mounir Benheddi mounir.benheddi@gmail.com
Camila Williamson camilaagw@gmail.com
Michal Kosinski kosinski.michal@gmail.com
Matej Petrik matej.petrik@unit8.com
Gaël Grosch† gael.grosch@unit8.com
† *Unit8 SA, Switzerland*

Editor: Alexandre Gramfort

Abstract

We present Darts[1], a Python machine learning library for time series, with a focus on forecasting. Darts offers a variety of models, from classics such as ARIMA to state-of-the-art deep neural networks. The emphasis of the library is on offering modern machine learning functionalities, such as supporting multidimensional series, fitting models on multiple series, training on large datasets, incorporating external data, ensembling models, and providing a rich support for probabilistic forecasting. At the same time, great care goes into the API design to make it user-friendly and easy to use. For instance, all models can be used using `fit()`/`predict()`, similar to scikit-learn (Pedregosa et al., 2011).

Keywords: time series, forecasting, machine learning, deep learning, Python

1. Introduction

Time series forecasting has numerous industrial and scientific applications in logistics, predictive maintenance, energy, manufacturing, agriculture, healthcare, sales, climate science, and many other domains. While classical methods such as ARIMA and Exponential

1. https://github.com/unit8co/darts

Smoothing often give good results (Hyndman and Athanasopoulos, 2018), machine learning is becoming a more attractive option to improve the models' representation power and scale to larger datasets and higher dimensionalities. In fact, it has recently been shown that pure ML-based approaches based on generic deep learning architectures can beat classical methods on a variety of tasks (Oreshkin et al., 2019; Flunkert et al., 2017).

Perhaps more important than sheer accuracy, the arrival of modern machine learning opens the opportunity to re-think the forecasting practices and software. For example, classical methods typically require training one model per time series, whereas ML models usually work best when trained on datasets containing large numbers of time series. This and other paradigm changes – such as better support for high-dimensional data, iterative training based on stochastic gradient descent, or the possibility to tailor loss functions for specific needs – require new tools and appropriate APIs. In particular, user-friendly and powerful APIs are important to make ML approaches as easy to use as classical techniques, which is necessary for larger-scale adoption by practitioners.

Several strong time series forecasting toolkits exist; however, they focus on classical methods or do not support deep learning and training models on multiple series (Hyndman and Khandakar, 2008; Jiang et al., 2021; Löning et al., 2019; Hosseini et al., 2021; Bhatnagar et al., 2021), or have lower-level APIs (Alexandrov et al., 2020; Beitner et al.). Darts proposes a new relatively high-level API unifying classical and ML-based forecasting models.

2. Design Principles and Main Features of Darts

2.1 Time Series Representation

Darts has its own `TimeSeries` data container type, which represents one time series. `TimeSeries` are immutable and provide guarantees that the data represents a well-formed time series with correct shape, type, and sorted time index. `TimeSeries` can be indexed either with Pandas `DatetimeIndex` or `RangeIndex` (Wes McKinney, 2010). The `TimeSeries` are wrapping around three-dimensional xarray `DataArray` (Hoyer and Hamman, 2017). The dimensions are (*time, component, sample*), where *component* represents the dimensions of multivariate series and *sample* represents samples of stochastic time series. The `TimeSeries` class provides several methods to convert to/from other common types, such as Pandas Dataframes or NumPy arrays (Harris et al., 2020). It can also perform convenient operations, such as math operations, indexing, splitting, time-differencing, interpolating, mapping functions, embedding timestamps, plotting, or computing marginal quantiles. For immutability, `TimeSeries` carry their own copy of the data and heavily rely on NumPy views for efficiently accessing the data without copying (e.g., when training models).

The main advantage of using a dedicated type offering such guarantees is that all models in Darts can consume and produce `TimeSeries`, which in turn helps to offer a consistent API. For instance, it is easy to have models consuming the outputs of other models.

2.2 Unified High-Level Forecasting API

All models in Darts support the same basic `fit(series: TimeSeries)` and `predict(n: int) -> TimeSeries` interface to be trained on a single series `series` and forecast `n` time steps after the end of the series. In addition, most models also provide richer functionalities; for instance the ability to be trained on a `Sequence` of `TimeSeries` (using calls like `fit([series1, series2, ...])`). Models can have different internal mechanics (e.g., sequence-to-sequence, fixed lengths, recurrent, auto-regressive), and this unified API makes it possible to seamlessly compare, backtest, and ensemble diverse models without having to know their inner workings.

Some of the models implemented in Darts at the time of writing are: (V)ARIMA, Exponential Smoothing, AutoARIMA (Smith et al., 2017), Theta (Assimakopoulos and Nikolopoulos, 2000), Prophet (Taylor and Letham, 2018), FFT-based forecasting, RNN models similar to DeepAR (Flunkert et al., 2017), N-BEATS (Oreshkin et al., 2019), TCN (Bai et al., 2018), TFT (Lim et al., 2021) and generic regression models that can wrap around any external tabular regression model (such as scikit-learn models). The list is constantly expanding and we welcome external and reference implementations of new models.

2.3 Training Models on Collections of Time Series

An important part of Darts is the support for training one model on a potentially large number of separate time series (Oreshkin et al., 2021). The `darts.utils.data` module contains various classes implementing different ways of slicing series (and potential covariates) into training samples. Darts selects a model-specific default slicing logic, but it can also be user-defined in a custom way if needed. All neural networks are implemented using PyTorch (Paszke et al., 2019) and support training and inference on GPUs . It is possible to consume large datasets that do not hold in memory by relying on custom `Sequence` implementations to load the data in a lazy fashion.

2.4 Support for Past and Future Covariates

Several models support *covariate* series as a way to specify external data potentially helpful for forecasting the target series. Darts differentiates *future covariates*, which are known into the future (such as weather forecasts) from *past covariates*, which are known only into the past. The models accept `past_covariates` and/or `future_covariates` arguments, which make it clear whether future values are required at inference time and reduces the risks to make mistakes. Covariate series need not be aligned with the targets, as the alignment is done by the slicing logic based on the respective time axes.

2.5 Probabilistic Forecasting

Some models (and all deep learning models) in Darts support probabilistic forecasting. The joint distributions over components and time are represented by storing Monte Carlo samples in the `TimeSeries` objects directly. This representation is very flexible as it does not rely on any parametric form and can capture arbitrary joint distributions. The computational cost of sampling is usually negligible, as samples can be efficiently computed in a vectorized way using batching. Probabilistic deep learning models can fit arbitrary likelihood forms, as long as the negative log-likelihood is differentiable. At the time of writing, Darts provides 17 distributions out-of-the-box (both continuous and discrete, univariate and multivariate). Finally, it offers a way to specify time-independent priors on the distributions' parameters, as a way to specify prior beliefs about the output distributions.

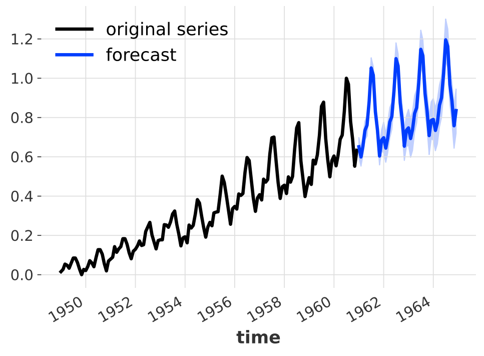
2.6 Other Features

Darts contains many additional features, such as transformers and pipelines for data pre-processing, backtesting (all models offer a `backtest()` method), hyperparameter search, extensive metrics, a dynamic time warping module (Berndt and Clifford, 1994), and ensemble models (with the possibility to use a regression model to learn the ensemble itself). Darts also contains *filtering* models such as Kalman filters and Gaussian Processes, which offer probabilistic modelling of time series. Finally, the `darts.datasets` module contains a variety of publicly available time series which can be conveniently loaded as `TimeSeries`.

3. Usage Example

The code below shows how to fit a single TCN model (Bai et al., 2018) with default hyper-parameters on two different (and completely distinct) series, and forecast one of them. The network outputs the parameters of a Laplace distribution. The code contains a complete predictive pipeline, from loading and preprocessing the data, to plotting the forecast with arbitrary quantiles (shown on the right).

```python
from darts.datasets import AirPassengersDataset, MonthlyMilkDataset
from darts.dataprocessing.transformers import Scaler
from darts.models import TCNModel
from darts.utils.likelihood_models import LaplaceLikelihood as LL

air = AirPassengersDataset().load()
milk = MonthlyMilkDataset().load()

scaler_air, scaler_milk = Scaler(), Scaler()
air_s = scaler_air.fit_transform(air)
milk_s = scaler_milk.fit_transform(milk)

model = TCNModel(input_chunk_length=24,
                 output_chunk_length=12,
                 likelihood=LL())
model.fit([air_s, milk_s], epochs=100)

pred = model.predict(n=48, series=air_s,
                     num_samples=500)

air_s.plot(label='original series')
pred.plot(low_quantile=.1, high_quantile=.9, label='forecast')
```



4. Conclusions

Darts is an attempt at democratizing modern machine learning forecasting approaches, and unify them (along with classical approaches) under a common user-friendly API. The library is still under active development, and some of the future work includes supporting static covariates and providing a collection of models pre-trained on large datasets, similar to what exists in the computer vision and NLP domains.

References

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. Turkmen, 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`.

V. Assimakopoulos and K. Nikolopoulos. The theta model: a decomposition approach to forecasting. *International Journal of Forecasting*, 16(4):521–530, 2000. ISSN 0169-2070. doi: https://doi.org/10.1016/S0169-2070(00)00066-2. URL `https://www.sciencedirect.com/science/article/pii/S0169207000000662`. The M3- Competition.

S. Bai, J. Z. Kolter, and V. Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *CoRR*, abs/1803.01271, 2018. URL `http://arxiv.org/abs/1803.01271`.

Beitner et al. Pytorch forecasting. `https://github.com/jdb78/pytorch-forecasting`. [Online; accessed 27-September-2021].

D. J. Berndt and J. Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, USA:, 1994.

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.

V. Flunkert, D. Salinas, and J. Gasthaus. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *CoRR*, abs/1704.04110, 2017. URL `http://arxiv.org/abs/1704.04110`.

C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, Sept. 2020. doi: 10.1038/s41586-020-2649-2. URL `https://doi.org/10.1038/s41586-020-2649-2`.

Hosseini et al. Linkedin greykite. `https://github.com/linkedin/greykite`, 2021. [Online; accessed 27-September-2021].

S. Hoyer and J. Hamman. xarray: N-D labeled arrays and datasets in Python. *Journal of Open Research Software*, 5(1), 2017. doi: 10.5334/jors.148. URL `http://doi.org/10.5334/jors.148`.

R. Hyndman and G. Athanasopoulos. *Forecasting: Principles and Practice.* OTexts, Australia, 2nd edition, 2018.

R. J. Hyndman and Y. Khandakar. Automatic time series forecasting: the forecast package for R. *Journal of Statistical Software*, 26(3):1–22, 2008. URL `https://www.jstatsoft.org/article/view/v027i03`.

Jiang et al. Facebook kats. `https://github.com/facebookresearch/Kats`, 2021. [Online; accessed 27-September-2021].

B. Lim, S. O. Arik, N. Loeff, and T. Pfister. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4):1748–1764, 2021. ISSN 0169-2070. doi: https://doi.org/10.1016/j.ijforecast.2021.03.012. URL `https://www.sciencedirect.com/science/article/pii/S0169207021000637`.

M. Löning, A. J. Bagnall, S. Ganesh, V. Kazakov, J. Lines, and F. J. Király. sktime: A unified interface for machine learning with time series. *CoRR*, abs/1909.07872, 2019. URL `http://arxiv.org/abs/1909.07872`.

B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio. N-BEATS: neural basis expansion analysis for interpretable time series forecasting. *CoRR*, abs/1905.10437, 2019. URL `http://arxiv.org/abs/1905.10437`.

B. N. Oreshkin, D. Carpov, N. Chapados, and Y. Bengio. Meta-learning framework with applications to zero-shot time-series forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(10):9242–9250, May 2021. URL `https://ojs.aaai.org/index.php/AAAI/article/view/17115`.

A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

T. G. Smith et al. pmdarima: Arima estimators for Python, 2017. URL `http://www.alkaline-ml.com/pmdarima`. [Online; accessed 27-September-2021].

S. J. Taylor and B. Letham. Forecasting at scale. *The American Statistician*, 72(1):37–45, 2018.

Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56 – 61, 2010. doi: 10.25080/Majora-92bf1922-00a.