BEYOND FIT & PREDICT: FORECASTING API FOR THE FOUNDATION MODEL ERA

Franz Király
sktime
German Center for Open Source AI
fkiraly@gcos.ai

Felipe Angelim
sktime
felipeangelim@pm.me

Simon Franz Albert Blanke
sktime
simon.blanke@protonmail.com

Benedikt Heidrich
sktime
benedikt.heidrich@sktime.net

Jigyasu
sktime
jigyasu@outlook.in

ABSTRACT

The emergence of Time Series Foundation Models (TSFMs) challenges the traditional `fit-predict` design pattern used in several forecasting libraries. Current frameworks force TSFM workflows—which distinguish between global pre-training, zero-shot inference, and fine-tuning—into interfaces originally designed for local, task-specific models. This mismatch necessitates workarounds that compromise evaluation integrity and leak data. We propose a formal API expansion to `sktime`, also applicable to other frameworks, that introduces a dedicated `pretrain` phase. This design restores command–query separation, keeps evaluation, ensembling, and deployment model-agnostic, and provides a unified interface for both classical and foundation models.

Track: Industry & Applications

1 THE SEMANTIC GAP IN FORECASTING APIS

Modern Machine Learning frameworks have been organized around the `fit-predict` paradigm (Pedregosa et al., 2011; Azul Garza, 2022; Löning et al., 2019; Herzen et al., 2022). This abstraction encodes a simple contract: `fit` consumes data to estimate parameters, and `predict` uses those fixed parameters to produce predictions. The pattern matches classical statistical forecasters (e.g., ARIMA) and many supervised learning approaches: training is a distinct phase, followed by inference.

Global models, such as Time Series Foundation Models (TSFMs), complicate this picture because they separate *global* learning from *local* task instantiation. A TSFM typically acquires broad capabilities during pre-training on large corpora. Later, at inference time, it may need to adapt to a particular series by conditioning on a context window or fine-tuning. These operations are not merely implementation details; they reflect a different lifecycle than the train-from-scratch assumption embedded in `fit-predict`.

When global forecasting workflows are expressed through the classic interface, a common workaround is to pass adaptation information through `predict` (e.g., by providing recent history or additional context in a way that influences internal state). This blurs the boundary between commands and queries (Meyer, 1997): `predict` is no longer a pure read-only operation and may introduce side effects. In practice, this can introduce subtle failure modes: silent data leakage if model parameters are updated using information beyond the intended training fold, fragile backtesting pipelines that must special-case model families, and reduced composability because downstream

tools cannot reliably assume that `predict` is side-effect free. Such problems, caused by software design, are symptoms of *abstraction debt* (Sculley et al., 2015).

The core issue is not simply that TSFMs are new models, but that the interface lacks an explicit separation of *global pre-training* and *local task adaptation*. Without a dedicated phase for global learning and controlled local adaptation, libraries either overload existing verbs or rely on conventions that are easy to violate. A robust design should make these stages' responsibilities explicit so that evaluation harnesses and composition operators can enforce clean information boundaries by construction.

2 Proposed Interface: The Explicit Pre-train Phase

We propose expanding the forecaster lifecycle by introducing an explicit `pretrain` verb and a corresponding state, and changing the main responsibility of `fit` from learning parameters to binding the model to a specific task. This maintains backward compatibility while formalizing the distinct stages of TSFM usage. The result is a uniform three-stage lifecycle that makes global learning, task binding, and inference explicit, allowing evaluation and composition utilities to rely on `predict` being side-effect free. Concretely, in `sktime` (Löning et al., 2019), the interface becomes:

1. **`pretrain(y, X=None)`:** A global learning phase. For TSFMs, this performs computationally expensive training on a large corpus or fine-tuning on domain-specific data. For classical statistical models, this is a no-op.
2. **`fit(y, X=None)`:** Local task instantiation. This binds the model to the target series, setting the forecasting cutoff and validating data schemas. Crucially, `fit` does *not* necessarily imply heavy training; for a zero-shot TSFM or a Tabular Foundation Model (Hollmann et al., 2022; Qu et al., 2025), it simply loads the context.
3. **`predict(fh, X=None)`:** Pure inference. Because adaptation is handled upstream, this method remains stateless and free of side effects, strictly mapping a horizon to a forecast.

3 Use Cases Enabled by a Uniform Three-Stage Contract

Making `pretrain`, `fit`, and `predict` explicit is not only a semantic refinement; it enables software patterns that otherwise require model-specific branching. Below we highlight some of the common operators that become straightforward once all forecasters adhere to the same lifecycle and `predict` remains a pure query.

Heterogeneous ensembling without bespoke adapters. Ensembling models with complementary inductive biases is a strong baseline—for example, combining a local statistical model that captures short-range autocorrelation with a global TSFM that captures cross-series structure. Today, mixing such families often requires bespoke adapters because some models train locally while others are used in a zero-shot manner. Under the proposed contract, classical models implement `pretrain` as a no-op while TSFMs may implement meaningful `pretrain`; both still follow `fit(y)` $\rightarrow$ `predict(fh)` at task time, enabling standard stacking or voting forecasters without model-specific integration logic.

Conformal prediction as a composable uncertainty layer. In many deployments, calibrated prediction intervals matter more than marginal improvements in point accuracy. Conformal methods can wrap any base forecaster by learning calibration statistics during `fit` (from permissible data only) and then augmenting the outputs during `predict` without ever accessing targets. This makes uncertainty portable across model classes and reduces the risk of accidental leakage.

Backtesting without model-type switches. A rolling-window evaluator can remain agnostic to whether a model is local or pre-trained: call `fit` on the training fold, then `predict(fh)` on the horizon. Evaluation harnesses do not need to special-case TSFMs, and the interface prevents passing targets into `predict`.

4 CONCLUSION

The rigid adherence to `fit-predict` is becoming a technical debt in the era of foundation and global models. By formalizing the `pretrain` phase and assigning clear responsibilities, we align the software architecture with the mathematical reality of modern learning workflows. This proposal offers a minimal surface-area change that restores safety, enables modular fine-tuning, and ensures that forecasting libraries remain robust infrastructure for both research and production.

REFERENCES

Cristian Challú Kin G. Olivares Azul Garza, Max Mergenthaler Canseco. StatsForecast: Lightning fast forecasting with statistical and econometric models. PyCon Salt Lake City, Utah, US 2022, 2022. URL https://github.com/Nixtla/statsforecast.

Julien Herzen, Francesco LÃ¤ssig, Samuele Giuliano Piazzetta, Thomas Neuer, LÃ©o Tafti, Guillaume Raille, Tomas Van Pottelbergh, Marek Pasieka, Andrzej Skrodzki, Nicolas Huguenin, Maxime Dumonal, Jan KoÅcisz, Dennis Bader, FrÃ©dÃ©rick Gusset, Mounir Benheddi, Camila Williamson, Michal Kosinski, Matej Petrik, and GaÃl 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.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. Tabpfn: A transformer that solves small tabular classification problems in a second. *arXiv preprint arXiv:2207.01848*, 2022.

Markus Löning, Anthony Bagnall, Sajaysurya Ganesh, Viktor Kazakov, Jason Lines, and Franz J Király. sktime: A unified interface for machine learning with time series. *arXiv preprint arXiv:1909.07872*, 2019.

Bertrand Meyer. *Object-oriented software construction*, volume 2. Prentice hall Englewood Cliffs, 1997.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

Jingang Qu, David HolzmÃžller, GaÃl Varoquaux, and Marine Le Morvan. Tabicl: A tabular foundation model for in-context learning on large data. *arXiv preprint arXiv:2502.05564*, 2025.

David Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. Hidden technical debt in machine learning systems. *Advances in neural information processing systems*, 28, 2015.