Short Term Time-Series Forecasting Framework

Gašper Kreft
gasper.kreft@gmail.com
Jožef Stefan Institute
Ljubljana, Slovenia

Aljaž Osojnik
aljaz.osojnik@ijs.si
Jožef Stefan Institute
Ljubljana, Slovenia

Martin Žnidaršič
martin.znidarsic@ijs.si
Jožef Stefan Institute
Ljubljana, Slovenia

Bernard Ženko
bernard.zenko@ijs.si
Jožef Stefan Institute
Ljubljana, Slovenia

Abstract

This paper presents a configurable and easy to use framework for short-term time-series forecasting that combines automated feature extraction from time-series and exogenous variables, direct multi-horizon model training and their evaluation. The framework includes standard machine learning algorithms and is designed for quick offline model prototype development and real-time prediction, rather than development of state-of-the-art models and their fine tuning. We demonstrate its practical use for forecasting electricity consumption of a commercial gas station. Its modular design allows the forecasting pipeline to be easily adapted to different time-series variables and external data sources and enables quick prototyping of forecasting models.

Keywords

short-term forecasting, machine learning, timeseries forecasting framework, direct multi-horizon forecasting, tsfresh

1 Introduction

Short-term forecasting is frequently used in energy management systems to support the planning and operation of energy resources, and is the key component of the *smart grid*. Individual installations may combine conventional loads with technologies such as heat pumps, water storage tanks, battery storage, and controllable HVAC systems. These components can produce supply and demand patterns that vary over time, and their forecasting enables better resource planning and operation.

Forecasting models for real-world applications face several practical constraints. Temporal patterns can occur at different time scales, and may depend on different external or *exogenous* variables such as weather or number of customers. A model intended for operational use must also be trained and evaluated in a manner that mimics its practical use, and we must ensure that evaluation procedure only uses information that would have been available at the prediction time. A forecasting approach therefore needs to combine multiple feature sources while allowing the model configuration to be adapted to different forecasting tasks.

This work presents a configurable short-term forecasting framework designed around these requirements. The framework supports automated feature engineering, direct multi-horizon prediction, integration of exogenous variables and their forecasts, and real-time inference within a common pipeline. Its application is demonstrated on a real-life problem of forecasting electricity consumption of a commercial gas station. The work was performed within the SEEDS Project[1], where it represents a component of an aggregation platform for providing electricity flexibility services.

The source code and example data used to demonstrate the framework are publicly available at https://github.com/gkreft01/ts_framework/.

2 Related Work

Recent developments in time-series forecasting have increasingly demonstrated the effectiveness of machine-learning methods for complex forecasting tasks. The M5 Accuracy competition at Kaggle [3] provides strong empirical evidence in this direction. Across the competition, which involved 42,840 hierarchical retail sales time-series and related exogenous variables, the top-performing methods were all based on machine learning. In particular, the 50 highest-ranked methods outperformed the strongest statistical benchmark, with LightGBM-based approaches achieving substantial improvements in forecasting accuracy. The authors consequently identified the superior accuracy of machine learning methods and the value of exogenous variables as two of the main findings of the competition [3].

Automated feature engineering provides another approach for improving machine learning based time-series forecasting. Wang et al. [6] investigated the use of automatically extracted time-series features within an automated machine learning framework for multi-step-ahead forecasting. Their approach constructs a large set of statistical time-series features using the *tsfresh* library and subsequently applies machine learning models to the resulting feature representation. The authors found that incorporating time-series features improved forecasting performance and that multi-output forecasting generally performed better than recursive forecasting under the same computational budget. Their work demonstrates the potential of automated feature engineering for multi-step forecasting; however, the framework focuses on features derived from the target time series and does not incorporate exogenous variables such as weather forecasts.

The usefulness of exogenous variables may also depend on the forecasting horizon. Perez Guillen and Milanović [2] investigated the influence of exogenous variables, including weather, economic activity, and demographic information, across different forecasting time scales using LSTM models and real UK electric grid data. Their results identified an inflection point between approximately 30 and 90 days, below which the inclusion of exogenous variables could reduce forecasting accuracy, while beyond this range they improved it.

[1] https://project-seeds.eu/

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
Information Society 2026, Ljubljana, Slovenia
© 2026 Copyright held by the owner/author(s).
https://doi.org/10.70314/is.2026.sikdd.100

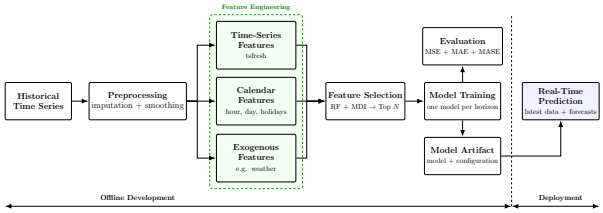


Figure 1: High-level overview of the proposed forecasting framework, showing different steps and components

The strategy used to generate multiple future predictions is an important consideration. Recursive forecasting trains a single one-step-ahead model and repeatedly feeds its predictions back into the model, which can lead to accumulated forecasting errors as the horizon increases. In contrast, direct forecasting trains separate models for the individual forecasting horizons, avoiding the propagation of previous prediction errors. Comparative studies in energy-demand forecasting and other real-world multi-step forecasting tasks have investigated these strategies using classic machine learning and deep learning models, demonstrating that the choice of forecasting strategy can substantially affect predictive accuracy, with direct and multiple-output approaches often outperforming recursive ones [4, 5].

Building on these findings, the proposed framework combines automated time-series feature extraction and selection, model training for direct multi-horizon forecasting, and model evaluation. It also enables easy deployment of the trained models. In contrast to approaches that rely only on information from the forecasted time-series, the framework is designed to also enable inclusion of exogenous variables such as weather. As such it is a useful tool for quick forecasting model prototyping in a range of time-series applications.

The work most similar to our is the AutoML approach for time-series proposed by [6]. Their pipeline is very similar to ours, but they use automatic search for all relevant hyperparameters, including window size, feature subset, learning algorithm and its hyperparameters. This approach is more complex and much more computationally expensive. It is therefore less appropriate for quick prototyping, though it is expected to produce more accurate forecasting models.

The following section describes the proposed framework, while Section 4 demonstrates its practical application. We conclude the paper by discussing insights gained from real-world implementation and suggesting directions for future work.

3 Methodology

The proposed forecasting framework is a configurable pipeline for multivariate time series forecasting and is implemented in *Python*. It consists of four stages: (1) data acquisition, (2) feature engineering, (3) feature selection, and (4) model configuration and training and (5) model evaluation. All components are parameterized through a unified configuration object (described in Section 4.1), allowing the same pipeline to be reused for different forecasting tasks by modifying only the experiment configuration. The trained model can also be easily deployed for real-time prediction. See Figure 1 for a high-level diagram.

3.1 Data Acquisition

The framework provides a flexible data acquisition layer for obtaining data from different sources. The data acquisition layer is intentionally separated from the remaining components of the framework, allowing users to adapt the database connections and data file acquisition procedures to their own data sources without affecting other components. We assume the time-series data points to be equidistant with the same time step.

3.2 Feature Engineering

3.2.1 Preprocessing. Prior to feature extraction and model training, missing timestamps are identified by comparing the input time series with a regular time index created with the defined time step, i.e., time step present in the data. Missing observations are then imputed using forward filling, carrying the most recently available value forward to preserve the regular temporal structure required by subsequent processing steps. Because non-aggregated time-series often include many spikes, we optionally smooth the data before continuing with the next steps.

3.2.2 Time-Series Features. Input time-series are transformed into a collection of statistical features using the *rolling window* approach and the *tsfresh* library [1]. The framework supports the extraction of features over multiple temporal windows to capture patterns occurring at different time scales. For each time-series, rolling windows of configurable lengths are used to represent both short-term and longer-term temporal behavior. The resulting features can therefore capture recent dynamics as well as patterns that develop over longer periods.

Prior to feature computation, differencing and moving-average smoothing can optionally be applied. Time-series can additionally be aggregated to a lower temporal resolution before rolling windows are applied. If this option is selected, features are computed for each rolling window and subsequently aligned back to the original sampling frequency using forward filling.

3.2.3 Calendar Features. In many problem domains the forecasted value depends on time of day, day of week or season of the year. The framework automatically derives features representing hour, weekday, weekend and month from the last timestep available at the time of forecasting. Additionally, it uses the *holidays* library to count the number of days until the next workday.

3.2.4 Exogenous Features. Forecasted variables are frequently affected by exogenous processes such as weather, for which we typically have forecasts provided by weather services. It is possible to include such forecasts and other exogenous variables as features in our framework. Available forecasts are transformed so that every prediction timestamp has future forecast values available for each forecasting horizon. During both training and testing, the framework uses archived numerical weather forecasts rather than observed meteorological measurements. This ensures that the model is trained under the same information constraints encountered during operational deployment, where only forecast weather is available. While this generally results in higher prediction errors due to the uncertainty introduced by meteorological forecasts, it provides a more realistic estimate of real-world forecasting performance and avoids optimistic bias.

3.3 Feature Selection

Using the *tsfresh* library combined with different time window sizes can result in a large number of features, many of which might be irrelevant to our forecasting problem. As a next step we therefore perform feature selection with the *Random Forest* algorithm, which we selected due to its efficiency and robustness, but it is straightforward to replace it with any feature selection

algorithm implemented in the *scikit-learn* library. We apply the algorithm for each forecasting horizon separately to compute the *Mean Decrease in Impurity (MDI)*, which is an estimate of feature importance. These feature importance scores are then averaged across all forecasting horizons, and the specified number of features with the highest mean importance is selected for training the final model.

3.4 Forecasting Model Training

The framework implements the direct multi-horizon model training process and trains a separate forecasting model for each prediction horizon using the selected training features and the corresponding target values. Again, we use the *Random Forest* algorithm for model training, due to its efficiency and robustness, but it can easily be replaced by some other algorithm from the *scikit-learn* library. The resulting multi-horizon model is as a model artifact together with the selected feature names, configuration, and training time. This allows the trained model to be subsequently loaded and evaluated or deployed for forecasting.

3.5 Evaluation

The trained model is evaluated on a held-out test set that is not used during model training. Before evaluation, the features required by the model (i.e., selected features before model training) are computed for the testing data. The model generates predictions for the complete forecasting horizon, which are then compared with the corresponding true target values. Performance is assessed using *Mean Squared Error (MSE)* and *Mean Absolute Error (MAE)*. In addition, *Mean Absolute Scaled Error (MASE)* is calculated by scaling the model's MAE against the MAE of a naïve forecast based on the value from the corresponding time step of the previous day. Lower values of all three metrics indicate better forecasting performance.

See Figure 2 for an illustration of testing the trained models on a test set. Red dotted rectangles represent the data from which the features were calculated, and the orange squares the predictions. Each x_i from the test set is first compared with the model's prediction $\hat{x}_{i,1}$ made one time step in advance then with the prediction $\hat{x}_{i,2}$ made two time steps in advance, and so on. In this context, we get a separate error measure for each horizon:

$$\mathrm{MAE}_h = \frac{1}{N-2m+1} \sum_{i=m}^{N-m} |x_{i+h} - \hat{x}_{i+h,h}|$$

$$\mathrm{MAE}_{\mathrm{naive},h} = \frac{1}{N-2m+1} \sum_{i=m}^{N-m} |x_{i+h} - x_{i-H}|$$

$$\mathrm{MASE} = \sum_{h=1}^{H} \frac{MAE_h}{MAE_{\mathrm{naive},h}},$$

where m is the size of the largest rolling window used for feature computation, N the test set size, and H the total number of horizons.

Finally, plots presenting predicted and true values on the test data are generated to provide a visual assessment of the model's forecasting behaviour.

3.6 Real-Time Prediction

The trained model can easily be deployed for real-time forecasting. The framework loads a previously trained model and retrieves the most recent data required for feature generation. The required historical period is determined automatically from the largest temporal window used by the model. Features are then generated using the same preprocessing and feature engineering procedures applied during training, and the most recent feature vector is passed to the trained model together with any required exogenous variables of forecasts. The model produces predictions for the configured forecasting horizon, which are assigned to the corresponding future timestamps. The resulting forecasts can then be stored in a database for subsequent use. This process allows the same trained model and feature engineering configuration to be used for operational forecasting.

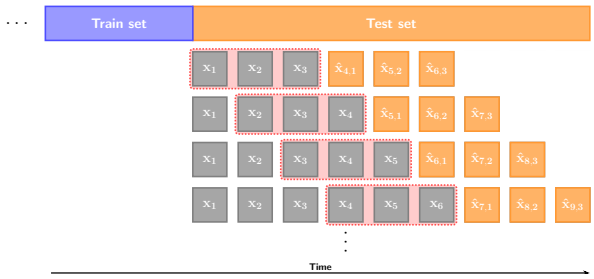


Figure 2: Illustration of the multi-horizon evaluation procedure. For each forecast origin, the red dotted rectangle denotes the measurements used for feature calculation, while the orange squares denote the resulting forecasts

4 Use Case Example

We demonstrate our tool on a task of forecasting electricity consumption of a single heat pump, which is a part of a pilot study within the SEEDS project. The data was collected by our project partner Petrol d.d., and comes from one of their commercial gas stations, while the weather data was collected from the Slovenian Environment Agency Web site. The use case scenario requires a forecasting model that generates one-day-ahead electricity consumption forecasts for a complete day in 15 min steps.

The example data set is provided in our repository (https://github.com/gkreft01/ts_framework/) as an SQLite database (file `example.db`). The database includes two tables, one with (normalized) heat pump electricity consumption (variable `Active_Power`) in 15 min steps for two months (ranging from 2026-03-01, 00:00h to 2026-05-01, 00:00h), and another with weather forecasts in 3 hour steps for the closest location (Jože Pučnik Airport), covering the same time interval. Weather data includes variables such as temperature, wind speed and precipitation.

4.1 Model Configuration and Training

The user needs to provide model configuration, which includes definition of the target variable, feature-extraction parameters, forecasting horizon, test period, model parameters, and optional external forecasts, These are defined through a `ModelConfig` object in file `experiments.py`.

In our example (see Listing 1), the target is `Active_Power`, sampled at 15 min steps, to which smoothing is applied with a window size 8 (`target_smooth_size`). Computation of time-series features is defined through method `FeatureSpec`, which specifies the rolling-window size, data resolution reduction (aggregation), generated feature set, output smoothing and differencing. In this configuration, two sets of features are computed, one with

rolling window size 96 and another with size 96*7, corresponding to one day and one week. The first set uses data at original 15 min resolution, while the second aggregates original 15 min data points into one daily value (i.e., 96 15 min steps). The third parameter determines whether *tsfresh* computes a bigger (`EfficientFCParameters`) or smaller (`MinimalFCParameters`) set of features, both sets use the bigger set (value `False`). The next parameter enables smoothing of output target predictions (used in the first set, where values are smoothed over 8 time steps). The last parameter enables differencing of data, but is not used in neither set (set to 0 or missing, instead of 1). The 50 highest-ranked features are retained (`top_n_features`), and a Random Forest model with 100 trees is used for direct multi-horizon forecasting. Temperature and accumulated precipitation forecasts (variables `temperature_c` and `precip_accumulated_mm`) are additionally specified as exogenous forecasts. The test set is defined as the last 14 values of the complete data set (`test_size`).

Listing 1: Model configuration example

```
HEAT_PUMP_CONFIG = ModelConfig(
    config_name="heat_pump",
    target_name="Active_Power",
    target_smooth_size=8,
    feature_params={
        "Active_Power": [
            FeatureSpec(96, 1, False, 8, 0),
            FeatureSpec(96*7, 96, False)
        ]
    },
    top_n_features=50,
    model_type=RandomForestRegressor,
    test_size=14,
    model_kwargs={
        "n_estimators": 100,
        "max_features": "log2",
        "random_state": 42
    },
    additional_forecasts={
        "Letališče Jožeta Pučnika Ljubljana":
        ["temperature_c", "precip_accumulated_mm"]
    }
)
```

Once model configuration (`experiments.py`) and database connection (`data_config.py`) are provided, the complete pipeline can be executed and model trained using a single command:

```
python pipeline.py --start-date 2026-03-01 --end-date
      2026-05-01 --config HEAT_PUMP_CONFIG
```

Results are stored in`model_runs.jsonl`, together with the current configuration parameters. We can now generate forecasts with:

```
python predict_realtime.py --config-name 'heat_pump' --
      forecast-date 2026-04-29
```

Scheduled calls of this command can provide us with real-time forecasts according to the model configuration. The forecasts are stored in a separate table in the database (`example.db`).

4.2 Forecast Results

Figure 3 illustrates the forecasts generated by the framework for the first day of the test set. The blue and red lines show the true and forecast energy consumption, respectively, while the dotted line represents the temperature forecast used as an exogenous input to the model. On the test set, the model achieved an MSE of 0.22, an MAE of 0.38, and a MASE of 0.90.

5 Conclusions

This paper presented a configurable framework for quick prototyping and deploying short-term time-series forecasting models.

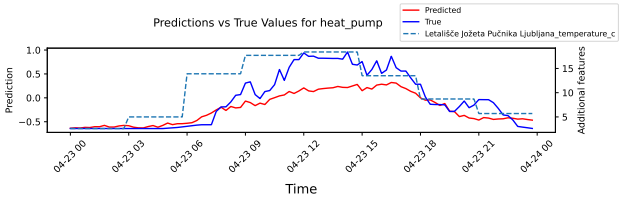


Figure 3: Example of heat pump forecasting for one day. The red line represents the predicted and the blue line true heat pump consumption; the dashed line is the temperature forecast used as an exogenous input

The framework combines automated feature extraction using *tsfresh* and feature selection, direct multi-horizon forecasting, and the integration of exogenous variables. Its modular architecture allows inclusion of different data sources, and simple adaptation to different forecasting applications.

The practical use case from the SEEDS project demonstrates how the tool can be easily configured and applied to a realistic forecasting scenario. The framework enables fast forecasting model prototyping and supports both offline evaluation and real-time prediction, using the same configuration.

6 Further Work

The current programmatic user interface can still be improved, but a suitable graphical user interface would further simplify configuration, model training, evaluation, and real-time forecasting, making the framework accessible to even wider audience.

Acknowledgments

The authors acknowledge the financial support from the European Union through the SEEDS project under the Horizon Europe Programme (GA No. 101138211) and the Slovenian Research and Innovation Agency (Research Core Funding No. P2-0103).

References

[1] Maximilian Christ, Nils Braun, Julius Neuffer, and Andreas W Kempa-Liehr. 2018. Time Series FeatuRe Extraction on basis of Scalable Hypothesis tests (tsfresh – A Python package). *Neurocomputing* 307 (Sept. 2018), 72–77.

[2] Airam Perez Guillen and Jovica V. Milanović. 2025. The Influence of Exogenous Variables on Accuracy of Demand Forecasting over Different Time Scales. In *2025 IEEE PES/IAS PowerAfrica*. IEEE, Cairo, Egypt, 1–5. doi:10.1109/powerafrica65840.2025.11289083

[3] Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. 2022. M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting* 38, 4 (Oct. 2022), 1346–1364. doi:10.1016/j.ijforecast.2021.11.013

[4] Erik Fernando Mendez-Garces, David Buldain, and María Paz Comech. 2026. Comparative Evaluation of Direct and Recursive Multi-Step Forecasting for Electricity Demand Using Deep Learning and Gradient Boosting Models. *Energies* 19, 15 (July 2026), 3563. doi:10.3390/en19153563

[5] Souhaib Ben Taieb, Gianluca Bontempi, Amir Atiya, and Antti Sorjamaa. 2011. A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition. doi:10.48550/ARXIV.1108.3259

[6] Can Wang, Mitra Baratchi, Thomas Bäck, Holger H. Hoos, Steffen Limmer, and Markus Olhofer. 2022. Towards Time-Series Feature Engineering in Automated Machine Learning for Multi-Step-Ahead Forecasting. *Engineering Proceedings* 18, 1 (2022), 1–11. doi:10.3390/engproc2022018017